UNIVERSIDAD CARLOS III DE MADRID

Ampliación del Entorno OWASP WebGoat

Ingeniería Técnica en Informática de Gestión
Proyecto Fin de Carrera

Alumno: Isaac Casanova Martínez
Tutor: José María de Fuentes García-Romero de Tejada
Co-tutor: Jorge Blasco Alís
Fecha: 4 de marzo de 2011

Índice de Contenidos

1.1 Contenido

Índice de Ilustraciones

Índice de Tablas

Agradecimientos

Este proyecto de fin de carrera supone el final de una etapa que no hubiese podido recorrer sin la ayuda, el apoyo y los consejos de muchas personas.

Por ello, quiero agradecer en primer lugar a mis padres, Alberto y Amparo, y a mi hermana Lucía, por haber estado siempre a mi lado apoyándome, dándome fuerzas y esperanzas en los momentos más duros, y por haber sabido ayudarme a escoger siempre el camino correcto.

Gracias Teresa Irene, por estar siempre fiel a mi lado, por tu cariño, por tu apoyo incondicional y por ser un ejemplo de superación incansable.

A José María, Teresa y Carlota por el cariño que me han dado estos últimos años, por sus muchos consejos siempre útiles y acertados y por mostrarme la fuerza que tenéis para superar cualquier dificultad que se presente, siempre con un trabajo incansable y dispuestos a ayudarme cuando lo he necesitado.

También quiero agradecer a mis amigos Marcos, Juanjo y Roberto que de una forma o de otra hayan estado a mi lado, en los buenos momentos pero también acompañándome en los difíciles.

Finalmente, tengo que agradecer a mi tutor de proyecto Don José María de Fuentes García-Romero de Tejada, por su paciencia, dedicación, comprensión y la confianza que ha depositado en mi persona para llevar a cabo este trabajo.

Muchas gracias a todos ellos, a mi familia y a todas aquellas personas que no he mencionado pero que de alguna forma me han acompañado hasta aquí.

Resumen

Los fallos de seguridad en aplicaciones software pueden causar funcionamientos erróneos no deseados, que causen daños económicos y jurídicos para los propietarios y usuarias de estas, puesto que gestionan información confidencial crítica que debe ser protegida.

Las aplicaciones residentes en entorno web, son accesibles por todas las máquinas que accedan a la red donde estén ubicadas (redes locales y/o Internet). Este factor aumenta el número de amenazas y vulnerabilidades que deberán tenerse en cuenta para elaborar las contramedidas a incorporar en el proceso de fabricación de software seguro.

WebGoat es una aplicación web de carácter educativo, cuyo objetivo es enseñar a sus usurarios algunas de las técnicas más comunes de ataques sobre aplicaciones web.

El presente proyecto tiene dos objetivos principales:

1. Implementar un sistema multilenguaje para que el contenido de las lecciones se pueda leer al menos en inglés (el idioma actual de la aplicación) y en castellano, a elección del usuario. Se implementará un sistema flexible para que la ampliación a otros idiomas sea fácil y rápido.
2. Implementar nuevas lecciones que completen el conjunto de las ya existentes. Para ello se realizará un estudio de las lecciones disponibles en WebGoat y de aquellas que aún no se han incluido.

Para llevar a cabo el proyecto se partirá de la versión 5.2 para desarrolladores de WebGoat, ya que cuenta con todas las lecciones implementadas en la versión 5.2 estándar y dispone de un entorno de Eclipse pre-configurado que facilita las modificaciones de código que exijan los objetivos del proyecto.

1 Introducción

Internet, web, correo electrónico, blogs, foros, comercio electrónico, son conceptos muy presentes en la sociedad del siglo XXI. Son herramientas de gestión del conocimiento que permiten obtener y difundir información en un plazo de tiempo muy corto.

Ahora bien, ¿qué es la información? Según indica la Real Academia Española (RAE) en una de sus acepciones, la información es la *comunicación o adquisición de conocimientos que permiten ampliar o precisar los que se poseen sobre una materia determinada*.

El ser humano se está sirviendo de Internet para informarse y ampliar sus conocimientos, además de difundir los suyos propios, pero ¿es aconsejable fiarse de toda la información publicada? ¿Es posible garantizar el buen uso de la información que nosotros suministramos a Internet?

Los responsables de dar respuesta a estas preguntas serán las personas encargadas de fabricar las aplicaciones que gestionan la información en Internet.

Este proyecto ha sido desarrollado para estas personas (estudiantes, ingenieros y programadores), con el fin de ayudarles a construir aplicaciones seguras que contengan una información veraz y fiable.

1.1 Estudio Preliminar

La seguridad en aplicaciones web es difícil de enseñar, ya que es complicado encontrar aplicaciones web que permitan explorar sus vulnerabilidades y sirvan como ejemplo para que los alumnos aprendan cómo funcionan los ataques y cómo defenderse de ellos.

WebGoat es una aplicación web de carácter educativo, formada por un conjunto de lecciones que ilustrarán a los alumnos sobre los fallos de seguridad en aplicaciones web.

Cada lección simulará una aplicación web deliberadamente vulnerable, para que el alumno, adquiriendo las nociones necesarias, consiga realizar el ataque y en algunos casos implementar una defensa para protegerse.

En la versión 5.2, WebGoat cuenta con más de 30 lecciones implementadas en un único idioma, el inglés.

En cada lección, el alumno podrá acceder a las siguientes secciones:

1. Escenario de simulación: Donde se incluirán los elementos con los que el alumno deberá interactuar para superar la lección. En esta sección también se incluirán unas breves instrucciones para informar al alumno del objetivo de la lección.
2. Planteamiento de la lección: Texto que contiene información relacionada con la lección, y que el alumno deberá aprender para poder superar la lección.
3. Solución de la lección: Texto que indicará al alumno los pasos a seguir para superar la lección.
4. Código Java: Contendrá el código fuente del fichero Java que genera la lección activa.
5. Ver Pistas: Mostrará textos de ayuda para que el usuario pueda superar la lección.
6. Ver Parámetros: Contendrá los últimos parámetros enviados al servidor por parte del navegador del alumno.
7. Ver Cookies: Mostrará las cookies enviadas al cliente por parte del servidor.

La Ilustración 1 muestra dónde se encuentran estas secciones dentro de la aplicación:

Ilustración 1: Secciones de cada lección de WebGoat

1.2 Objetivos del Presente Proyecto

Dado el escenario descrito en el apartado anterior, se plantean varios objetivos para el proyecto que se agruparán en dos:

1. Convertir WebGoat en multilenguaje: en la actualidad el sistema está orientado para soportar un único idioma, pero se pretende aumentar el radio de accesibilidad al mismo superando las barreras del lenguaje. Se deberá ampliar el sistema actual para que permita poder ver las lecciones en dos o más idiomas distintos.
2. Crear nuevas lecciones: Es de vital importancia que WebGoat esté lo más actualizado posible, para que la aparición de nuevas formas de ataque no dejen obsoleto el sistema. Para ello se implementarán tres nuevas lecciones:
    a. Variante de Clickjacking: Técnica que consiste en engañar al usuario incitándole a realizar acciones pensando que está haciendo otras.
    b. Escaneo Anónimo de Puertos mediante CSPP[1]: Enseña un método de obtener información de servidores de bases de datos, utilizando el formulario de acceso a los gestores de las propias bases de datos
    c. Repudiation Attack: Técnica para engañar a sistemas vulnerables, haciendo que las acciones registradas en el mismo sean erróneas y lleven a confusión.

1.3 Organización del Presente Documento

Para una mejor comprensión de los temas que se tratarán en este proyecto, se realizará una explicación de la estructura seguida para contar de la forma más clara posible los objetivos pretendidos, y cómo se ha trabajado para alcanzarlos.

El documento estará estructurado en las siguientes secciones:

1. Introducción: Ayudará a comprender los factores de necesidad para la elaboración de este proyecto y el contexto en el que se ubica.
2. Análisis: Contendrá todos los aspectos relativos al planteamiento de las necesidades que provocan la elaboración del proyecto
3. Diseño: Contendrá las soluciones lógicas aplicadas para satisfacer las necesidades expuestas en el apartado de análisis.
4. Implementación e Implantación del Software: Incluirá los aspectos técnicos de las soluciones explicadas en el apartado de diseño.

[1] CSPP: Connection String Parameter Pollution (Contaminación de parámetros en la cadena de conexión)

5. Implicaciones legales del proyecto
6. Conclusiones y líneas futuras: Contendrá las interpretaciones del autor sobre los resultados obtenidos al finalizar del proyecto.
7. Bibliografía y Referencias: Incluirá los nombres fuentes que han servido de ayuda y apoyo en la realización del proyecto.
8. ANEXO I Gestión del Proyecto: Informará sobre los métodos utilizados para planificar y abordar el presente proyecto.
9. ANEXOII Glosario de Términos: Contendrá un conjunto de términos que aparecen a lo largo del presente documento con una pequeña descripción.

2 Análisis

El contenido de análisis abarca el estudio del proyecto a realizar y de los factores que de alguna forma, intervienen o intervendrán en su implementación y explotación.

Para una mejor comprensión, se explicará cada punto partiendo de una visión general, para posteriormente profundizar en cada una de las nuevas lecciones a implementar:

1. En la sección "Introducción a WebGoat" se explicará qué es WebGoat, quienes son sus creadores, cuál es su objetivo y cuál es su entorno de trabajo.
2. La sección "Análisis de las nuevas lecciones a implementar" presentará las nuevas lecciones a implementar.
3. En la sección "Arquitectura Preliminar y Análisis de las Tecnologías Impuestas" se abordará la estructura interna de WebGoat, las tecnologías que se han utilizado para su elaboración y las distintas alternativas que se pueden tomar para elaborar el presente proyecto
4. "Arquitectura Definitiva y Selección de Tecnologías" es la sección que explican la estructura, tecnología, plataforma y lenguaje que se va a utilizar para elaborar el proyecto.
5. La sección "Diagrama de Casos de Uso" presentará el comportamiento que deberá tener el sistema.
6. En la sección "Catálogo de Requisitos Software" se abordarán todas las necesidades que debe cubrir el software implementado.

2.1 Introducción a WebGoat

WebGoat (OWASP, WebGoat Project) es una aplicación web, lo que conlleva ceñirse a una estructura cliente servidor, en la cual el cliente realiza peticiones al servidor y este responde una serie de datos procesados (HTML, imágenes, Javascript, XML…).

La Ilustración 2 muestra el funcionamiento básico de esta estructura, en el que el cliente escribe en la barra de direcciones del explorador la dirección o URL de la lección 1 de WebGoat. El servidor procesa la petición y genera el código de la lección 1 de WebGoat, que será interpretado y procesado por el navegador del cliente:

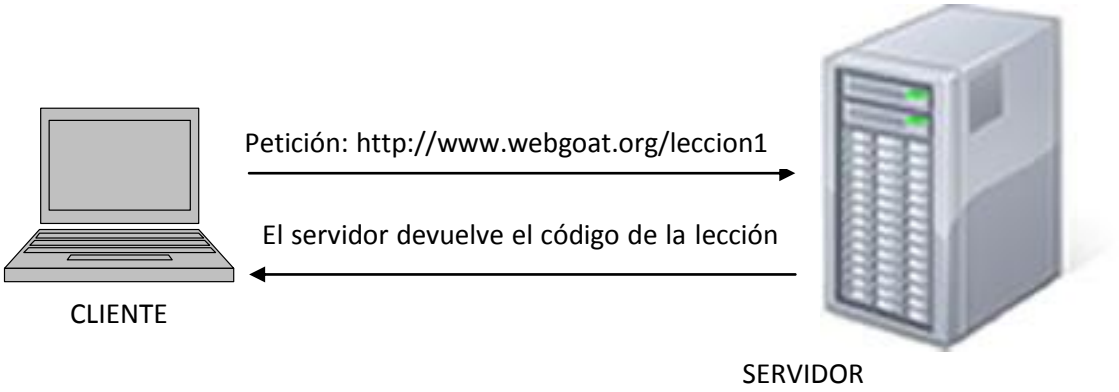

Ilustración 2 Arquitectura Básica Aplicación Web

Esta estructura requiere identificar qué código se ejecuta en el servidor y qué parte se ejecuta en el cliente:

1. Código ejecutado en el servidor: es aquel que se procesa para generar una respuesta a la petición del cliente. Este código está formado por scripts programados en lenguajes (Java, PHP, Visual Basic .NET…) que viene determinado por la plataforma (J2EE, LAMP, .NET …) instalada en el servidor.
2. Código ejecutado en el cliente: es generado por el servidor como respuesta a una petición del propio cliente. Este código es interpretado por el navegador (Internet Explorer, Google Chrome, Mozilla Firefox…), y puede contener diversos elementos: código HTML, código Javascript, imágenes, hojas de estilo, flash …

WebGoat es una aplicación creada y mantenida por OWASP[2] (OWASP, Open Web Application Security Project, 2001), una organización sin ánimo de lucro formada por una comunidad abierta y libre a nivel mundial, cuyo objetivo es investigar y mejorar la seguridad en las aplicaciones web.

Actualmente existen dos versiones estables de WebGoat:

1. WebGoat OWASP Standard 5.2: Incluye la versión 5.2 de WebGoat, el entorno de ejecución de Java (JRE) y un servidor Tomcat 5.5 (The Apache Software Foundation) ya configurado.
2. WebGoat OWASP Developer 5.2: contiene lo mismo que la versión estándar y además contiene un entorno de Eclipse pre-configurado, que permite crear nuevas lecciones y ampliar las funcionalidades de WebGoat.

[2] Proyecto de código Abierto de Seguridad en Aplicaciones Web, por sus siglas en inglés

La versión 5.3 de WebGoat está disponible, pero carece de documentación adecuada de ayuda para su instalación y funcionamiento. Además no existe una versión para programadores, lo que hubiera complicado la realización del proyecto.

2.2 Análisis de las Nuevas Lecciones a Implementar

En este apartado se definirán los conceptos de las nuevas lecciones a implementar, con el objeto de dar a conocer al lector qué es lo que se pretende realizar y el alcance de cada lección.

2.2.1 Clickjacking

Esta lección trata de enseñar en qué consiste la técnica de una variante de Clickjacking, una forma de engañar al usuario haciéndole creer que está realizando acciones que tienen consecuencias desconocidas para él.

Originalmente el Clickjacking consiste en robar los clics que el usuario realiza con el ratón sobre una pantalla de un explorador de Internet. Un ejemplo claro de este tipo de ataque sería la captura del número secreto de acceso a la cuenta de un banco (en el caso de que el usuario introdujese los dígitos de acceso pulsando sobre la pantalla). El usuario malicioso modificaría la web original de forma que añadiría un *iframe*[3] transparente (con opacidad 0) justo encima del panel de dígitos para introducir la clave. El objetivo de esta capa transparente sería capturar la posición donde se realiza el clic de ratón para averiguar qué digito se ha pulsado, de forma que el usuario piensa que está pulsando sobre el panel cuando realmente está pulsando sobre la capa transparente.

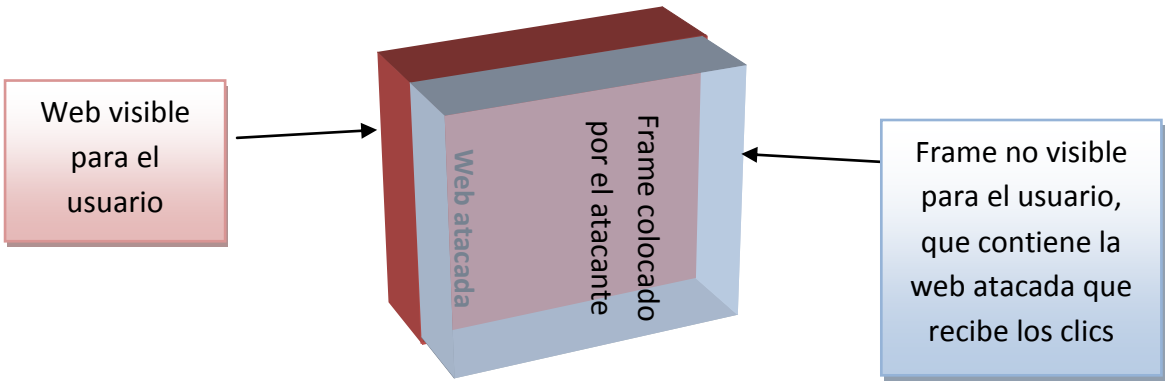

Ilustración 3 Método Clásico de Clickjacking

La variante de Clickjacking que se va a tratar realiza un ataque justo de la manera inversa, puesto que consistirá en cargar una página Web externa dentro de un *iframe* invisible (con opacidad 0, por ejemplo), y poner debajo del *iframe* invisible (con menor z-index), elementos HTML para que el usuario haga clic, pero que realmente tienen el *iframe* invisible encima, y es quien recibe el clic.

[3] Fragmento de código que permite definir una región que ocupará una parte de la pantalla en una página web.

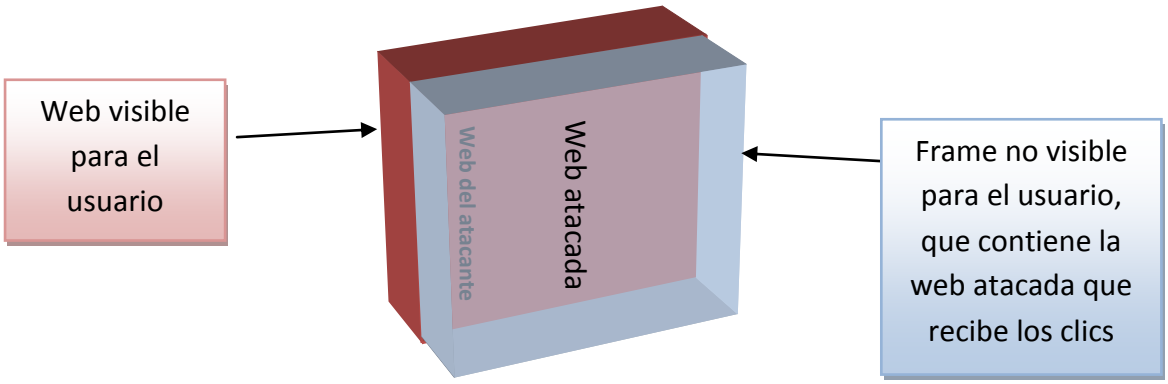

Ilustración 4 Método Alternativo de Clickjacking

Este ataque es lanzado por desarrolladores de páginas webs maliciosas, con el objetivo de capturar las pulsaciones de ratón que se hacen sobre los botones de la web maliciosa y emplearlos para votar noticias, pujas por eBay etc.…

Una forma de protegerse del Clickjacking sobre una página web es evitando que esta se pueda cargar en un *iframe*.

(Facebook, 2010) Recientemente se han detectado varios ataques de este tipo en la red social Facebook. Usan los comentarios en los muros para añadir fotos llamativas, estas fotos son un enlace con la web maliciosa, donde se pide al usuario que haga clic en un botón azul.

Este botón azul es en realidad un “Compartir” o un “Me gusta” en la página de perfil en Facebook de la víctima, con lo que sin saberlo está permitiendo que dicho contenido sea publicado en su cuenta, tentando a sus amigos a caer en la misma estafa.

Al usuario se le redirige a un vídeo de YouTube, creando de esta forma la ilusión de que al pulsar el botón se ha redirigido directamente a este vídeo.

2.2.2 Escaneo Anónimo de Puertos usando CSPP (Connection String Parameter Pollution)

Esta lección trata de enseñar cómo comprobar el estado de los puertos de un servidor de base de datos utilizando las vulnerabilidades de las cadenas de conexión.

En el ámbito de internet un puerto, es el valor que se usa, en el modelo de la capa de transporte, para distinguir entre las múltiples aplicaciones que se pueden conectar al mismo host (equipo o dispositivo). Aunque muchos de los puertos se asignan de manera arbitraria, ciertos puertos se asignan, por convenio, a ciertas aplicaciones particulares o servicios de carácter universal. De hecho, la IANA (Internet Assigned Numbers Authority (IANA)) determina, las asignaciones de todos los puertos

comprendidos entre los valores [0, 1023]. Por ejemplo, el servicio de conexión remota telnet, usado en Internet se asocia al puerto 23.

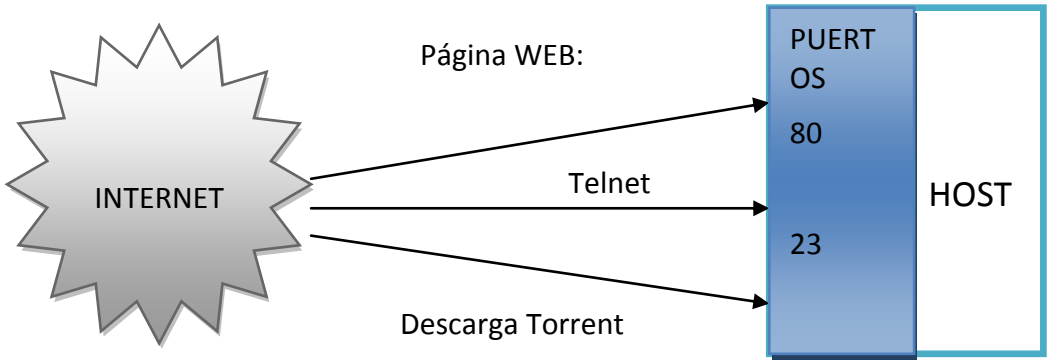

Ilustración 5Acceso a Puertos de un Host

Una cadena de conexión es una cadena de texto formado por pares variable - valor, separado por un signo de igual ("="), y cada par separado por punto y coma (";"). Cada par define una propiedad para especificar la conexión con la base de datos.

Una aplicación que desea conectarse a una base de datos utiliza la cadena de conexión para indicar (entre otros parámetros) el servidor donde se encuentra dicha base de datos y la forma en que se va a conectar a dicha base de datos (modo de autenticación, nombre de usuario, contraseña etc.)

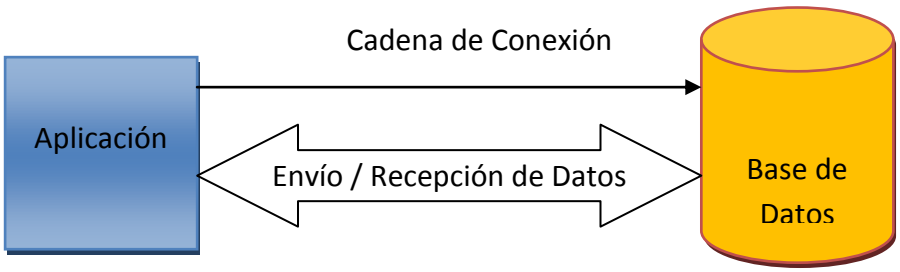

Ilustración 6 Comunicación Básica de una Aplicación con la Base de Datos

Las propiedades más comunes usadas en una cadena de conexión son las siguientes (ConnectionStrings.com):

1. ***Provider***: Establece el nombre del proveedor para la conexión.
2. ***Connection Timeout***: Establece el tiempo en segundos para esperar a una conexión antes de terminar el intento y generar una excepción, por omisión toma valor 15.
3. ***Initial Catalog***: Especifica el nombre de la base de datos, si se omite se utiliza la predeterminada del usuario.
4. ***Data Source***: Indica el nombre del servidor, en el caso de Access es el nombre de archivo.
5. ***Password***: Establece la contraseña del usuario.
6. ***User ID***: El identificador del usuario.
7. ***Integrated Security***: Establece el mecanismo de autenticación con el servidor, los valores posibles son TRUE y FALSE.
8. ***Persist Security***: Cuando se establece a FALSE, la información que requiere un alto grado de seguridad como la contraseña no se muestra una vez se haya establecido la conexión, por omisión toma valor FALSE.

Un atacante, utilizando la funcionalidad de configurar el puerto en la cadena de conexión podría utilizar una aplicación vulnerable a esta técnica para escanear servidores.

Consiste en realizar intentos de conexión por los diferentes puertos y ver los mensajes de error que se obtienen en cada una de las peticiones.

Estos intentos de conexión se realizan mediante inyecciones CSPP, que consisten en contaminar los parámetros de la cadena de conexión.

El ataque se realiza indicando el número de puerto después de la dirección de servidor separado por una coma y dentro, todo ello, de la inyección de la cadena de conexión.

El resultado del escaneo de puerto nos ofrecerá información relativa a ese puerto, indicando si está abierto, cerrado o protegido por un cortafuegos. Esta información nos sirve para detectar qué servicios comunes está ofreciendo la máquina y posibles vulnerabilidades de seguridad según los puertos abiertos. Es posible incluso detectar el sistema operativo utilizado por el servidor según los puertos que tenga abiertos.

2.2.3 Repudiation Attack

Esta lección trata de enseñar cómo realizar un ataque para burlar el seguimiento de las acciones que se le puede hacer a un usuario mediante *logs*[4].

Este tipo de ataque consiste en modificar los datos del usuario enviados a la aplicación, alterando así el rastro que puede dejar dicho usuario al realizar acciones dentro de la misma.

Este método le puede resultar útil a usuarios maliciosos que quieran realizar ciertas acciones y atribuírselas a otro usuario que puede existir o no en la aplicación.

Para realizar este ataque necesitaremos un analizador de redes que nos permita interceptar las peticiones enviadas al servidor y modificar los datos relativos a usuario, fechas de modificación y todos aquellos datos que sirvan para rastrear las acciones realizadas por un usuario.

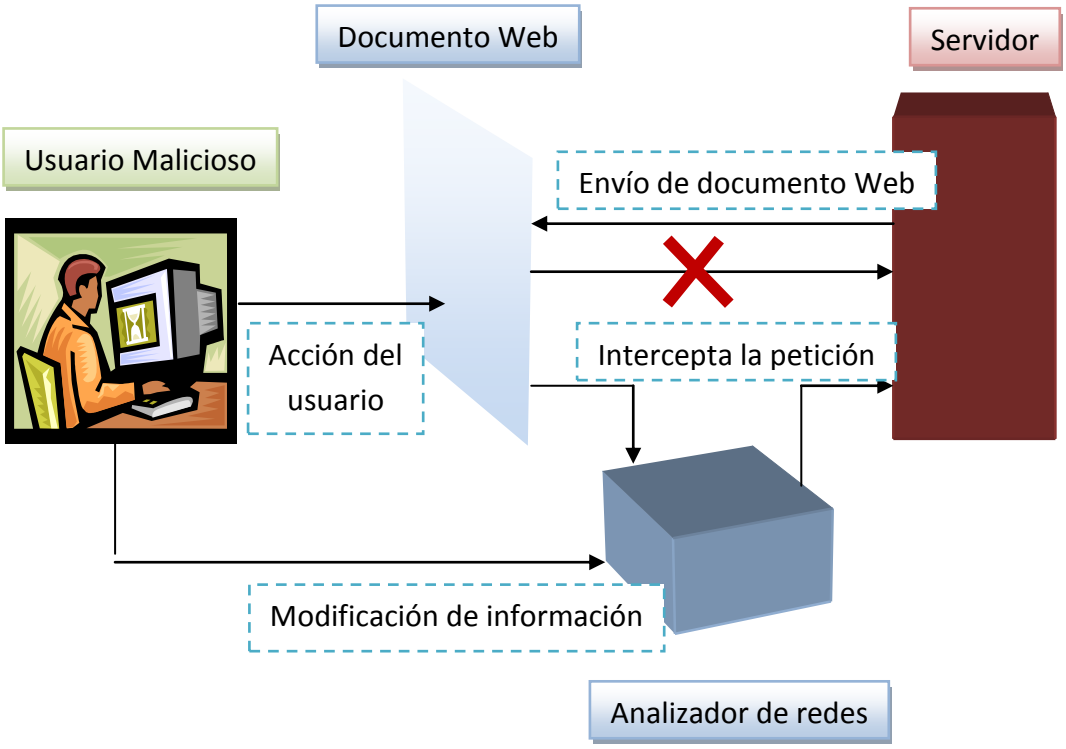

Ilustración 7 Esquema de Funcionamiento de Analizador de redes

El servidor envía al explorador del usuario un documento web que contiene un formulario con campos ocultos con información relativa al usuario para que el servidor sepa qué usuario pretende realizar qué cosa, al recibir la petición.

Si el usuario utiliza un analizador de redes para interceptar la petición antes de que llegue al servidor, podrá modificar el valor de los campos ocultos del formulario y confundir al servidor.

[4] Registro para almacenar las acciones que se realizan en un sistema.

Si el script ejecutado en el servidor almacena los datos de trazabilidad de acciones de los usuarios, almacenará datos erróneos, bien para que no se pueda averiguar qué usuario ha realizado la acción (si se usan datos de un usuario inventado por el atacante) o bien para asignar la acción a otro usuario (utilizando los datos de otro usuario registrado en el sistema).

2.3 Arquitectura Preliminar y Análisis de las Tecnologías Impuestas

2.3.1 Plataforma de desarrollo

WebGoat está implementado sobre la plataforma J2EE[5] (Oracle Corporation), lo que implica ciertas restricciones para desarrollar el código que ejecuta en el servidor. A continuación se explica en qué es y en qué consiste esta plataforma.

La plataforma J2EE define un estándar para el desarrollo de aplicaciones empresariales. La base de esta plataforma consiste en dividir las aplicaciones en múltiples módulos (fragmentos de código) que son o pueden ser usados en otras aplicaciones, proporcionando así un conjunto de servicios que permiten al programador cierto grado de abstracción y simplificando el desarrollo de software.

Esta plataforma aprovecha muchas de las características de J2SE (Java 2, Standard Edition), como son la portabilidad (un mismo módulo que puede ser usado en múltiples aplicaciones), la API JDBC (para acceso a Bases de datos), la tecnología CORBA para la interacción con los recursos existentes en la empresa y un modelo de seguridad que protege los datos incluso en aplicaciones web.

Las características que aporta J2EE son un soporte completo para componentes de Enterprise JavaBeans (que permite encapsular varios objetos simples en uno más complejo, pero que facilita su uso), API para Servlets de Java (paquete de recursos para el desarrollo de aplicaciones web, que abarca el manejo de sesiones, utilidades HTTP, empaquetado y seguridad a nivel de aplicación etc.), JavaServer Pages (tecnología que permite generar contenido dinámico para aplicaciones web) y la tecnología XML (permite compartir datos con otras aplicaciones independientemente de su plataforma).

2.3.2 Servidor

J2EE necesita un servidor Web que atienda las peticiones y gestione las respuestas generadas por la plataforma. WebGoat usa para ello un servidor Tomcat, un software de código abierto implementado para las tecnologías JavaServlet y JavaServer Pages.

[5] Java 2 Enterprise Edition

2.3.3 Almacenamiento de Datos

WebGoat no precisa de almacenar gran cantidad de datos, puesto que se trata de una aplicación de carácter informativo y educacional.

Por tanto no precisa de un gestor de base de datos de gran envergadura como pueden ser Oracle o SQLServer.

Los desarrolladores de WebGoat decidieron, por tanto utilizar dos tecnologías de almacenamiento de datos:

1. HSQLDB[6] (The hsql Development Group) Es un gestor de base de datos relacional implementado en Java.
2. XML: Esta tecnología es usada frecuentemente para el intercambio de datos entre varias tecnologías. Puede ser usado también para almacenar documentos con un formato definido por un DTD (documento que contiene información sobre la estructura de datos XML). Para acceder a los datos se pueden usar varias técnicas (SAX, DOM, JDOM …), el uso de cada técnica depende de cómo la aplicación quiere recuperar los datos.

2.3.4 Arquitectura Preliminar de Lecciones WebGoat

WebGoat está formado por un conjunto de módulos llamados lecciones, que obedecen a una misma estructura lógica que se muestra en la Ilustración 8:

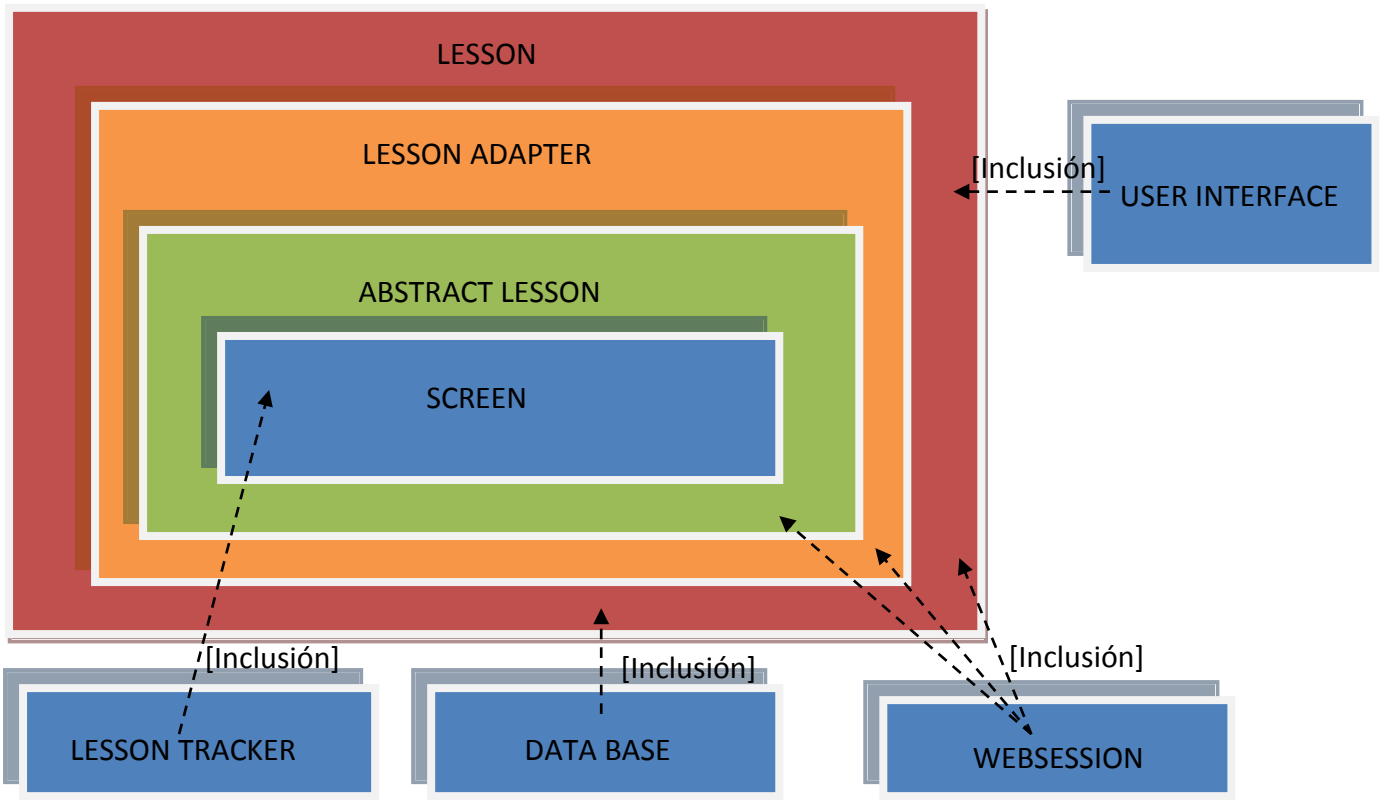

Ilustración 8 Esquema Conceptual de Arquitectura de las Lecciones

[6] HSQLDB: HyperSQL Data Base

Cada módulo o lección está formado por un conjunto de clases relacionadas entre sí y representadas por un rectángulo de color en la Ilustración 8.

Estas relaciones son de dos tipos:

1. ***Extensión***: Son las relaciones en las cuales una clase adopta las propiedades y métodos protegidos y públicos de otra y además añade sus propiedades púbicas y privadas. Esta relación se refleja en la Ilustración 8 por la clase *Lesson* que extiende la clase *Lesson Adapter*, que a su vez extiende la clase *Abstract Lesson*, y que a su vez extiende la clase *Screen*.
2. ***Inclusión***: Son aquellas en las que una clase puede utilizar los métodos y propiedades públicas de otra. En la Ilustración 8, se refleja este tipo de relación con las clases Lesson Tracker (incluida en la clase *screen*), *Websession* (incluida en las clases Abstract Lesson, Lesson Adapter y Lesson), *Database* (incluida en Lesson) y *User Interface* (Incluida en Lesson).

Las clases que forman la lección se exponen en la Tabla 1:

Tabla 1: Componentes de lección

Nombre	Descripción
Screen	Clase que contiene los métodos y propiedades que generan el código HTML que será enviado al usuario.
Abstract Lesson	Contiene métodos y propiedades comunes a todas las lecciones.
Lesson Adapter	Añade métodos y propiedades comunes a todas las lecciones y actúa de interface con la clase que define cada lección (en la Ilustración 8 es la Lección Final).
Lesson	Clase que contiene los métodos y propiedades propios de cada lección. Es la clase que se debe generar para crear una nueva lección.
Websession	Gestiona las variables de sesión del usuario.
LessonTracker	Controla las lecciones que el usuario ha superado.
Data base	Base de Datos donde se guardan los datos de la lección, si procede.
User Interface	Muestra al usuario los datos de la lección.

2.4 Arquitectura Definitiva y Selección de Tecnologías

Como se ha comentado en puntos anteriores la tecnología utilizada para este proyecto viene impuesta (plataforma J2EE), ya que el objetivo del mismo consiste en completar un desarrollo software existente y estable.

Pero además de contemplar esta opción, también se contempló que las nuevas lecciones se podrían haber implementado en otra tecnología como LAMP / WAMP o .NET con bases de datos SQLServer o MySQL, realizando una comunicación con la aplicación actual mediante Webservices por ejemplo, pero ello implicaría una mayor complejidad en todos los sentidos, ya que habría que instalar el nuevo entorno y emplear un tiempo de desarrollo en implementar módulos que ya existen en la plataforma actual.

La única decisión que se ha tomado con respecto a las tecnologías es la forma de almacenar el contenido multilenguaje de las lecciones. Las posibles soluciones que se plantearon fueron las siguientes:

1. **Almacenarlo en la base de datos HSQLDB** ya existente para almacenar datos de usuarios y algunas lecciones. Como ventaja de esta opción se diría que al ya estar definida la interfaz para comunicarse con otra base de datos nos ahorraríamos tiempo de implementación, pero en contra está que el tiempo de acceso a datos podría ser superior a otras opciones, además el tiempo que se emplearía en guardar todos los textos en los distintos idiomas sería muy elevado.
2. **Utilizar otro gestor de base de datos** como Oracle, SQLServer o MySQL. Estos gestores están muy optimizados y proveen de una gran cantidad de herramientas que mejoran la comunicación con la base de datos con respecto a HSQLDB, pero su implantación supondría un coste elevado de tiempos para el poco uso que se le darían a todos los recursos que ofrecen.
3. **Utilizar documentos XML**. Este sistema se basa en la generación de documentos de lenguaje de marcado que siguen un formato dado por un DTD (Documento de definición de tipos de datos). Un lenguaje de marcado es aquel formado por etiquetas (marcas que delimitan el comienzo y el final de una entidad) que estructuran el documento en un conjunto de elementos (en nuestro caso registros) delimitados por dichas etiquetas. Cada elemento deberá tener un conjunto de atributos que lo diferencien del resto de las entidades que forman el documento XML. La mayoría de los lenguajes de programación disponen de alguna herramienta que permita leer los datos de un documento XML para ser procesados y obtener la información necesaria.

De las tres soluciones, la última sería la más apropiada, ya que el tiempo de implementación sería mínimo, porque bastaría con crear un DTD y un documento XML por cada idioma. Además, una vez se tenga creado el documento en un idioma, sería muy fácil obtenerlo en cualquier otro, ya que bastaría con pasar el documento XML por un traductor de textos fiable y que pueda obviar las etiquetas que delimitan los elementos. El acceso a los datos es aceptable utilizando SAX (Saxproject) como intérprete de XML, ya que lee el documento secuencialmente hasta encontrar el elemento buscado.

Atendiendo a la arquitectura y teniendo en cuenta las nuevas especificaciones propuestas para el proyecto, podemos representar el nuevo WebGoat gráficamente mediante el diagrama de componentes UML mostrado en la Ilustración 9:

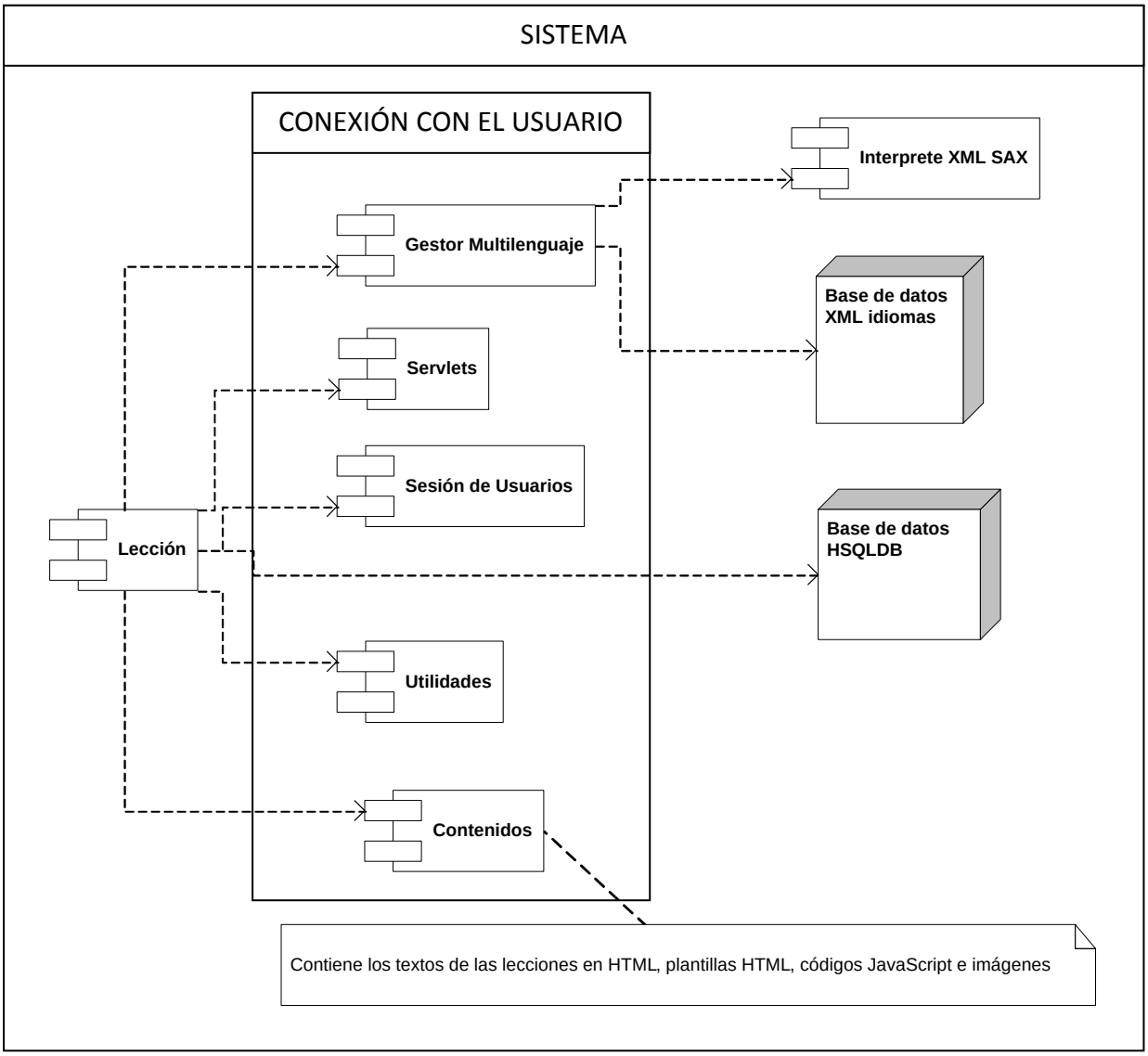

Ilustración 9 Diagrama de Componentes del Sistema

Como se ha comentado con anterioridad WebGoat está formado por lecciones.

Para esta nueva versión, se pide que cada una de estas lecciones sea multilenguaje, por tanto necesitaremos un componente que cargue el idioma seleccionado por el usuario para cada lección. Necesitaremos guardar los textos de las lecciones en los distintos idiomas en una base de datos XML y un intérprete adecuado para interpretarla.

Además, las lecciones dependen de otros componentes como servlets para proveer de contenidos a la lección, un componente que controla la sesión de los usuarios, otro con utilidades comunes a todas las lecciones y por último un componente de contenidos, que abarca las plantillas y contenidos HTML de las lecciones, así como los Javascript e imágenes que se mostrarán al usuario en la página mostrada en su navegador.

En la Tabla 2 se definen cada uno de los componentes de la Ilustración 9:

Tabla 2: Componentes del Sistema

NOMBRE	DESCRIPCIÓN
Lección	Abarca todos los componentes usados por el sistema para crear cada lección
Gestor Multilenguaje	Encargado de gestionar el idioma en el que se muestra el contenido de las lecciones
Servlets	Encargado de gestionar las peticiones del usuario y las respuestas del servidor
Sesión de Usuarios	Gestiona las sesiones de los usuarios que acceden al sistema y proporciona datos de los mismos a las lecciones
Utilidades	Conjunto de herramientas utilizadas y compartidas por las lecciones
Contenidos	Contenidos de las lecciones (imágenes, estilos, plantillas HTML, código Javascript…)
Intérprete XML SAX	Componente que procesa los documentos XML y devuelve la información precisada por la lección.
Base de datos de Idiomas	Base de datos en XML donde se almacenarán los textos explicativos de las lecciones en los distintos idiomas.
HSQLDB	Base de datos utilizada por algunas lecciones para generar escenarios ficticios.

2.5 Diagrama de Casos de Uso

En este apartado se describe el comportamiento del nuevo sistema en su interacción con el usuario.

Los casos de uso de la ampliación de WebGoat serían los indicados en la Ilustración 10

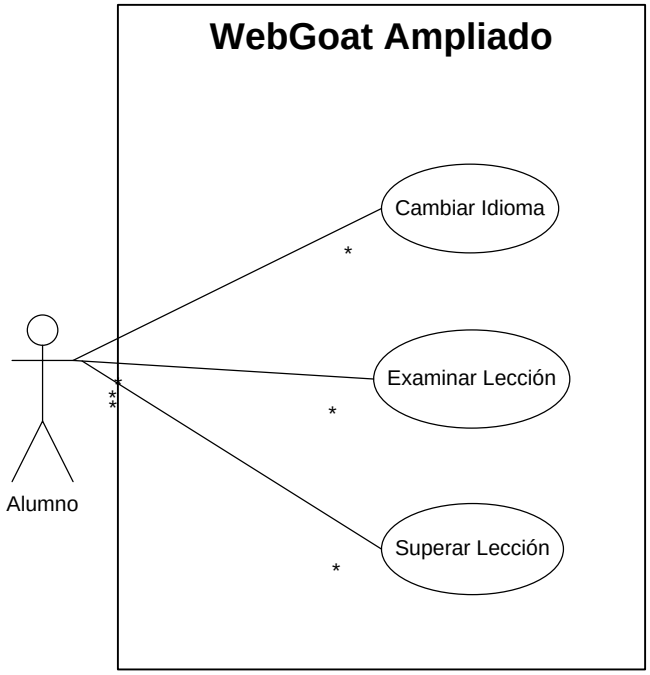

Ilustración 10 Diagrama de Casos de Uso

En la Tabla 3 se explica cada uno de los elementos del diagrama:

Tabla 3: Actores del Sistema

Actores del Sistema	
Alumno	Es el único actor que va a participar en el sistema. El sistema contiene un alumno por defecto, pero ofrece la posibilidad de añadir más de uno. El inconveniente de añadir más de un actor al sistema es que existen lecciones que piden al alumno que modifique el código Java que se ejecuta en el servidor, por lo tanto dicho código quedará modificado para todos los alumnos dados de alta en la misma aplicación.

Tabla 4: Caso de Uso - Cambiar Idioma

CASO DE USO		
Título: Cambiar Idioma		
Identificador: CU1	**Versión: 1**	**Fuente: Isaac Casanova Martínez**
Actores: Alumno		
Objetivo: El alumno seleccionará el idioma en el que quiere leer el contenido de las lecciones		
Precondiciones: El usuario deberá haberse registrado para modificar el idioma		
Postcondiciones: Las lecciones se mostrarán en el idioma seleccionado		
ESCENARIO BÁSICO		
1. Desplegar la lista de idiomas **2. Seleccionar idioma.**		
ESCENARIOS ALTERNATIVOS		
No Aplica		

Tabla 5: Caso de Uso - Examinar Contenido Lección

CASO DE USO		
Título: Examinar Contenido Lección		
Identificador: CU2	**Versión: 1**	**Fuente: Isaac Casanova Martínez**
Actores: Alumno		
Objetivo: El alumno seleccionará la lección que desea aprender y leerá los textos de su contenido.		
Precondiciones: El usuario deberá haberse registrado y seleccionar un idioma antes de seleccionar una lección y examinar su contenido		
Postcondiciones: El usuario habrá leído el contenido de la lección		
ESCENARIO BÁSICO		
1. Seleccionar lección **2. Leer instrucciones**		
ESCENARIOS ALTERNATIVOS		
ESCENARIO ALTERNATIVO 1		
1. Seleccionar lección 2. Leer Lesson Plan		
ESCENARIO ALTERNATIVO 2		
1. Seleccionar lección 2. Leer Solución		
ESCENARIO ALTERNATIVO 3		
1. Seleccionar lección 2. Ver Pistas		
ESCENARIO ALTERNATIVO 4		
1. Seleccionar lección 2. Ver código java		
ESCENARIO ALTERNATIVO 5		
1. Seleccionar lección 2. Ver cookies		
ESCENARIO ALTERNATIVO 6		
1. Seleccionar lección 2. Ver parámetros		

Tabla 6: Caso de Uso - Superar Lección

CASO DE USO		
Título: Superar Lección		
Identificador: CU3	**Versión: 1**	**Fuente: Isaac Casanova Martínez**
Actores: Alumno		
Objetivo: Superar la lección seleccionada por el usuario		
Precondiciones: El usuario deberá haberse registrado en el sistema y haber seleccionado una lección		
Postcondiciones: El sistema marcará la lección como superada		
ESCENARIO BÁSICO		
1. Simular Clickjacking **2. Implementar defensa contra Clickjacking** **3. Validar código de defensa implementado**		
ESCENARIOS ALTERNATIVOS		
ESCENARIO ALTERNATIVO 2		
1. Realizar inyección mediante CSPP para escanear puertos **2. Implementar defensa contra inyecciones CSPP** **3. Validar código de defensa implementado**		
ESCENARIO ALTERNATIVO 3		
1. Borrar un registro **2. Borrar otro registro haciéndose pasar por otro usuario**		

2.6 Catálogo de Requisitos de Software

Esta sección contendrá las especificaciones que determinarán el comportamiento de la ampliación de WebGoat.

La especificación de requisitos se realizará según establece el estándar ESA Lite (ESA Comité de Estandarización y Control de Software, 2003), clasificándolos por su tipología como se indica a continuación:

1. Requisitos Funcionales
2. Requisitos de Interfaz
3. Requisitos de Recursos
4. Requisitos de Seguridad
5. Requisitos de Mantenimiento

Cada tipo de requisito se corresponderá con una subsección que contendrá una breve descripción global de los requisitos que lo componen y un listado detallado de los mismos.

En cada sección, los requisitos se dispondrán en una tabla en forma de lista numerada para su identificación, seguimiento, trazabilidad y validación.

Para cada requisito debe completarse una fila que formará parte de una tabla siguiendo el formato indicado en la Tabla 7:

Tabla 7: Estructura Tabla Requisitos

[TIPOLOGÍA DE REQUISITO]					
Tipo:					
Id	**Ver.**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**

La estructura de la Tabla 7 se describe a continuación:

1. Primera fila: Contendrá el nombre de la tipología de los requisitos que formarán parte de la tabla. Solo se tendrán en cuenta los Requisitos de Software
2. Segunda Fila: Informará el tipo de requisitos que va a contener la tabla (Requisitos de Capacidad, Requisitos de Restricción, Requisitos Funcionales, Requisitos de Interfaz …)
3. Tercera Fila: Indicará el nombre de cada campo de los requisitos:
    a. Id: Identificador unívoco del requisito. Estará formado por un código alfabético que determinará la tipología y el tipo de requisito (será el mismo para todos los requisitos incluidos en el tipo) y por un código numérico que identificará cada requisito dentro de su tipo. A modo de ejemplo: RURC01 significará que se trata de un Requisito de Usuario (RU) del tipo Requisito de Capacidad (RC) colocado en la lista en primera posición (01).
    b. Prueba: Código de prueba, que relaciona el requisito con la prueba realizada para su verificación.
    c. Título: Nombre para identificar el requisito.
    d. Descripción: Texto explicativo del requisito.
    e. Estabilidad: Indican la probabilidad de que el requisito cambie a lo largo del desarrollo del proyecto
    f. Prioridad: Indica la importancia del requisito. Puede ser Alta, Media o Baja.

2.6.1 Requisitos Funcionales

Para una mejor comprensión de la especificación de requisitos, este apartado contendrá cuatro sub-secciones. La primera de ellas hará referencia a los requisitos de software funcionales para la aplicación en general, y las tres restantes se corresponderán con los requisitos funcionales de cada una de las nuevas lecciones a implementar.

REQUISITOS DE SOFTWARE DEL SISTEMA					
Tipo: Requisitos Funcionales					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRF01	PRF01	Selección de idioma	En la parte superior de la pantalla de bienvenida se deberá incorporar un campo desplegable que contenga los idiomas en los que están traducidas las lecciones de WebGoat. En este proyecto los idiomas contemplados serán español e inglés. Para cambiar de idioma el usuario deberá seleccionar dicho idioma del desplegable, y automáticamente los contenidos se mostrarán traducidos a este idioma.	Media	Alta
RSRF02	PRF02	Selección de lección	Los enlaces a las lecciones se encontrarán en el menú vertical de la izquierda, agrupados en secciones. Este menú estará formado por secciones, y al pulsar sobre cada una de ellas, se desplegará un listado de todas las lecciones que contenga. Al pulsar sobre la lección, esta se cargará en la zona de contenido de lecciones en el idioma seleccionado.	Media	Alta

Tabla 8: Requisitos de Software del Sistema - Requisitos Funcionales

REQUISITOS DE SOFTWARE PARA CLICKJACKING					
Tipo: Requisitos Funcionales					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRF03	PRF03	Acceso a Clickjacking	Clickjacking estará incluida en la sección "General" del menú de lecciones.	Baja	Media
RSRF05	PRF04	Simulación de Clickjacking	Se situará una capa no visible para el usuario sobre un botón que tenga como texto "Púlsame". El clic del usuario nunca debe llegar al botón, lo debe recibir la capa transparente	Alta	Alta
RSRF06	PRF05	Activar y Desactivar Clickjacking	Deberá existir un campo de tipo checkbox que permita activar y desactivar la simulación de Clickjacking. Cuando el Clickjacking esté desactivado el clic del usuario lo recibirá el botón.	Media	Media
RSRF07	PRF06	Pulsación del botón "Púlsame"	Al recibir el clic, se debe mostrar un mensaje al usuario informando de la acción.	Baja	Baja
RSRF08	PRF07, PRF08	Botón "Validar Código"	Para superar la lección el usuario deberá realizar modificaciones en el código fuente. Para verificar estas validaciones, se dispondrá de un botón que permita aplicar dichos cambios sobre la lección y que el usuario compruebe sus efectos.	Alta	Alta
RSRF09	PRF09	Superar Clickjacking	La lección se marcará como superada cuando el alumno consiga mostrar el contenido de la capa no visible. El sistema deberá detectarlo añadiendo un botón dentro de la capa no visible que el usuario deberá pulsar. Una vez pulsado, la lección se marcará como superada en el menú de lecciones.	Alta	Alta

Tabla 9:Requisitos Funcionales - Clickjacking

REQUISITOS DE SOFTWARE PARA ESCANEO ANÓNIMO DE PUERTOS CON CSPP					
Tipo: Requisitos Funcionales					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRF10	PRF10	Acceso a Escaneo de Puertos mediante CSPP	Escaneo de Puertos mediante CSPP estará incluida en la sección "General" del menú de lecciones.	Baja	Media
RSRF11	PRF11, PRF12, PRF13	Simulación de Escaneo de Puertos mediante CSPP	Se presentará un formulario que simule un control de acceso a un gestor de base de datos vía web. El formulario constará de dos campos: Nombre de Usuario y Password. Si no se introduce una inyección se mostrará un mensaje de usuario y contraseña incorrectos	Alta	Alta
RSRF12	PRF14	Botón "Validar Código"	Recargará la página actual, para actualizar el código que el alumno haya podido modificar para intentar superar la lección	Alta	Alta
RSRF13	PRF15	Superar Lección	Se deberá detectar que el alumno ha modificado el código fuente del fichero *antiCSPP.js* para que no se permita introducir inyecciones en la cadena de conexión.	Alta	Alta

Tabla 10: Requisitos Funcionales - Escaneo Anónimo de Puertos por CSPP

REQUISITOS DE SOFTWARE PARA REPUDIATION ATTACK					
Tipo: Requisitos Funcionales					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRF13	PRF13	Acceso a Repudiation Attack	Repudiation Attack estará incluida en la sección "General" del menú de lecciones.	Baja	Media
RSRF14	PRF14	Borrar Registro	Se mostrará una tabla con registros que el usuario pueda eliminar. Cuando el usuario elimina un registro se envía el nombre del propio usuario junto con la orden de eliminación para que la aplicación sepa quién realiza la acción.	Media	Alta
RSRF15	PRF15	Superar Lección de Repudiation Attack.	La lección se marcará como superada cuando la aplicación detecte que el nombre de usuario que le llega es distinto del usuario activo (El nombre del usuario activo siempre será "guest")	Alta	Alta

Tabla 11: Requisitos Funcionales - Repudiation Attack

2.6.1.1.1 Requisitos de Interfaz

Al igual que el apartado anterior, este contendrá cuatro sub-secciones, para exponer por separado los requisitos que afectan a todas las lecciones y los requisitos de las nuevas lecciones a implementar.

Tabla 12: Requisitos de Software - Requisitos de Interfaz

REQUISITOS DE SOFTWARE				
Tipo: Requisitos de Interfaz				
Id	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRI01	Estructura de interfaz	Todas las pantallas que forman parte de la aplicación guardarán una misma estructura definida al final de esta sección	Media	Media
RSRI02	Estructura del menú	El menú agrupará las lecciones en categorías. Las nuevas lecciones se incluirán en la categoría “General”	Baja	Baja
RSRI03	Contenido de la lección	Esta sección albergará las instrucciones y elementos de cada lección con los que el usuario debe interactuar para aprender y superarla. Deberá situarse en la parte inferior derecha de la pantalla	Alta	Media
RSRI04	Pistas de lección	Se mostrarán en la parte superior del contenido de la lección en color rojo	Alta	Media
RSRI05	Parámetros	Mostrará los parámetros pasados en la última conexión realizada con el servidor. Se mostrarán en la parte superior del contenido en color rojo	Alta	Media
RSRI06	Cookies	Se mostrará en la parte superior del contenido de la lección en color rojo	Alta	Media
RSRI07	Lesson Plan	Se mostrará en una capa sobre el contenido de la lección. Dicha capa tendrá un borde negro de 1 pixel de grosor. El tipo de letra será “Verdana” con tamaño de 10 pixeles.	Media	Alta
RSRI08	Título de Lesson Plan	Se mostrará centrado en la parte superior de la capa de Lesson Plan, con el formato: **Título de la Lección:** nombre_leccion. Donde nombre_leccion será el nombre de la lección activa. El texto de “Título de la Lección” deberá ir en negrita	Media	Baja
RSRI09	Título de Sección de Lesson Plan	Estará situado a la izquierda y en negrita.	Baja	Baja
RSRI10	Contenido de Sección de Lesson Plan	Se mostrará alineado a la izquierda dentro del contenedor de LessonPlan.	Baja	Baja

RSRI11	Solución de la lección	Se mostrará en una nueva ventana del explorador de Internet. El tipo de letra será "Verdana" con tamaño de 10 pixeles.	Alta	Alta
RSRI12	Título de la Solución	Se mostrará alineado a la izquierda, en la parte superior de la nueva ventana, con el formato: **Título de la Lección:** nombre_leccion. Donde nombre_leccion será el nombre de la lección activa.	Media	Media
RSRI13	Título de Sección de la Solución	Estará situado a la izquierda y en negrita.	Media	Media
RSRI14	Contenido de Sección de la Solución	Se mostrará alineado a la izquierda.	Baja	Media
RSRI15	Código Java de la Lección	Se mostrará como texto de solo lectura en una nueva ventana del explorador.	Baja	Media

A modo de ejemplo, todas las pantallas que forman parte de WebGoat deberán mantener la estructura indicada en la Ilustración 11:

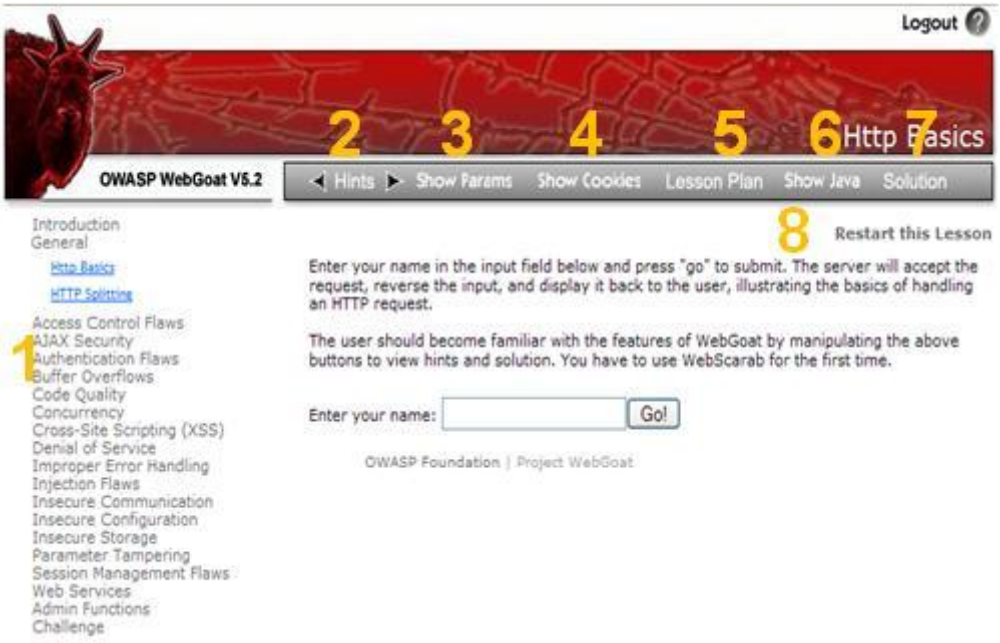

Ilustración 11 Estructura de Interfaz de Usuario de WebGoat

La Tabla 13 explicará el contenido de la Ilustración 11:

Tabla 13: Estructura de Interfaz de Usuario

Código	Descripción
1	El menú para navegar por las distintas lecciones. Está dividido en secciones que clasifican las lecciones por tipos de ataque.
2	El botón "Hints" permite al usuario ver las pistas que la lección ofrece para poder

Código	Descripción
	superarla.
3	El botón “Show params” muestra al usuario los parámetros pasados por GET.
4	“Show Cookies” muestra al usuario las cookies almacenadas en su equipo por WebGoat.
5	El botón “Lesson Plan” muestra un popup al usuario con toda la información de la lección.
6	“Show Java” muestra el código Java de la lección.
7	El botón “Solution” muestra en un popup la forma de superar la lección.
8	Contiene una breve introducción a la lección junto con unas pequeñas instrucciones y los elementos (formularios, tablas, botones …) necesarios para intentar superar la lección.

Tabla 14: Requisito de Interfaz - Clickjacking

REQUISITOS DE SOFTWARE PARA CLICKJACKING					
Tipo: Requisitos de Interfaz					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRI16	PRI16	Interfaz de Clickjacking	Se mostrará un escenario para simular un ataque de Clickjacking, que estará formado por: ➢ Botón con texto “Púlsame” que incite al usuario a pulsarlo. ➢ *Iframe* transparente situado sobre el botón “Púlsame”. Este *Iframe* tendrá opacidad 0 para evitar su visibilidad. ➢ Checkbox para activar y desactivar el *iframe*. ➢ Botón Validar Código. Se mostrará esto botón en la parte inferior junto con un texto explicativo de su funcionalidad. Estos elementos se mostrarán bajo las instrucciones de la lección, dentro del área de contenido.	Alta	Alta
RSRI17	PRI17	Capa oculta de Clickjacking	La capa oculta, en un principio, será invisible para el usuario. Pero internamente estará apuntando a una página interna que contendrá los siguientes elementos: ➢ Mensaje informativo que indique al usuario que ha superado la lección Botón de confirmación, que debe pulsar el usuario para confirmar que ha superado la lección.	Alta	Alta

Tabla 15: Requisito de Interfaz - Escaneo Anónimo de Puertos CSPP

REQUISITOS DE SOFTWARE PARA ESCANEO ANÓNIMO DE PUERTOS POR CSPP					
Tipo: Requisitos de Interfaz					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRI18	PRI18	Interfaz principal para Escaneo anónimo de puertos por CSPP	Esta lección contendrá un formulario que simule el control de acceso a una base de datos. Este formulario se mostrará centrado en la pantalla después de las instrucciones de la lección, y con un borde negro de 2 pixeles de ancho que delimite su tamaño. Está formado por tres elementos: ➢ Título del formulario: será "Database Access", estará centrado con el formulario, con letra negrita y con el fondo en color "coral" de CSS. ➢ Usuario: Campo de tipo texto donde el usuario introducirá su nombre de acceso. ➢ Password: Campo de tipo password, donde el usuario introducirá la clave de acceso a la base de datos. En la parte inferior de la lección se deberá incorporar un botón de "Validar Código".	Alta	Alta

Tabla 16: Requisitos de Interfaz: Repudiation Attack

REQUISITOS DE SOFTWARE PARA REPUDIATION ATTACK					
Tipo: Requisitos de Interfaz					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRI19	PRI19	Interfaz Repudiation Attack	Esta lección contendrá una tabla que simulará un listado de inventario de productos. Esta tabla está formada por filas de tres campos cada una: ➢ Producto: Contendrá el nombre del producto ➢ Precio: Contendrá el precio del producto Eliminar: Contendrá un aspa roja para eliminar el producto.	Alta	Alta

2.6.1.1.2 Requisitos de Recursos

Tabla 17: Requisitos de Software - Recursos

REQUISITOS DE SOFTWARE					
Tipo: Requisitos de Recursos					
Id	**Prueba**	**Título**	**Descripción**	**Estabilidad**	**Prioridad**
RSRH01	PRH01	Hardware del Sistema	Al tratarse de una aplicación web, se necesitará un equipo cliente que realice las peticiones e intérprete las respuestas, y un servidor que atienda las peticiones y las responda. El equipo cliente y el servidor serán físicamente el mismo, por lo que no será necesario montar ninguna infraestructura de red.	Media	Media
RSRH02	PRH02	Software del Cliente	El lado de cliente precisa de un explorador de internet, preferiblemente Mozilla Firefox versión 3.6. El explorador enviará las peticiones al servidor e interpretará las respuestas ofrecidas por este, para mostrárselas al usuario de una forma legible.	Alta	Alta
RSRH03	PRH03	Software del Servidor	Es necesario un software de servidor que atienda las peticiones del cliente, las procese y envíe la respuesta. Para ello WebGoat usa Tomcat versión 3.5.	Alta	Alta

2.6.1.1.3 Requisitos de Seguridad

No aplica

2.6.1.1.4 Requisitos de Mantenimiento

No aplica.

3 Diseño Detallado

En las secciones contenidas en este apartado se informará de las decisiones tomadas para dar la solución a los requisitos expresados en el apartado anterior, así como la forma en la que la solución va a ser desarrollada.

3.1 Diseño del Software

El contenido de este apartado se corresponderá con el diagrama de componentes expuesto en el apartado de "Análisis", sección "Arquitectura Definitiva y Selección de Tecnologías".

Se descompondrán todos los componentes en no más de dos niveles, con el objetivo de expresar con claridad la solución propuesta para satisfacer los requisitos especificados en el apartado anterior.

3.1.1 Diseño de Componente: Lección

Abarcará todos los elementos relativos a una lección de WebGoat, y estará formado por el diagrama de clases mostrado en la Ilustración 12:

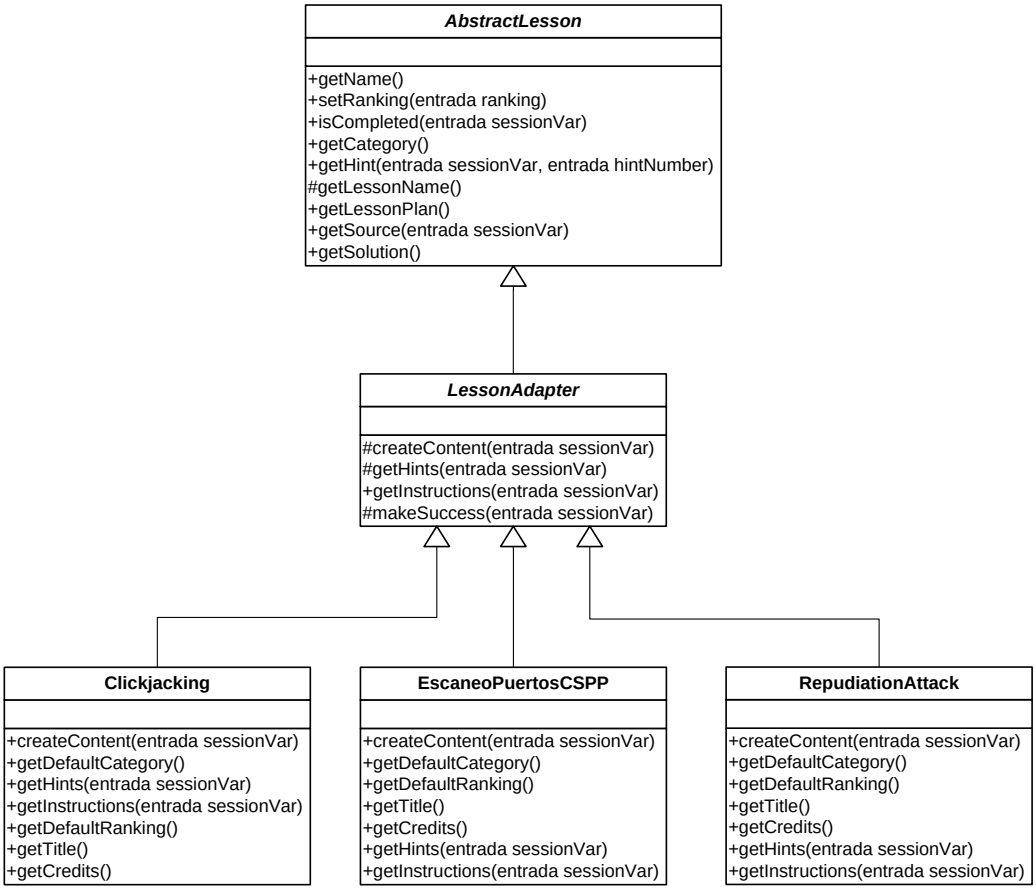

Ilustración 12 Diagrama de Clases del Componente Lección

Todas las lecciones de WebGoat tendrán una estructura común representada por las clases AbstractLesson y LessonAdapter.

Todos aquellos aspectos que diferencian unas lecciones de otras estarán contenidos en la clase propia de cada lección. En la Ilustración 12 únicamente están representadas las clases de las nuevas lecciones a implementar, pero todas las lecciones de WebGoat existentes siguen la misma estructura.

En la Tabla 18 se detallan cada una de las clases del componente Lección:

Tabla 18: Clases del componente Lección

NOMBRE	VISIBILIDAD	TIPO	DESCRIPCIÓN
AbstractLesson	Pública	Abstracta	Contendrá todos los elementos que puede necesitar una lección de WebGoat.
LessonAdapter	Pública	Abstracta	Amplía y completa la clase AbstractLesson añadiendo y modificando métodos y propiedades.
Clickjacking	Pública	Instanciable	Amplía y completa la clase LessonAdapter para implementar las funcionalidades que se le piden a la lección de Clickjacking.
EscaneoPuertosCSPP	Pública	Instanciable	Amplía y completa la clase LessonAdapter para implementar las funcionalidades que se le piden a la lección de Escaneo Anónimo de Puertos mediante CSPP.
RepudiationAttack	Pública	Instanciable	Amplía y completa la clase LessonAdapter para implementar las funcionalidades que se le piden a la lección de Repudiation Attack.

Los principales métodos de cada clase se describirán a continuación clasificados en cinco tablas, una por cada clase:

3.1.1.1.1 AbstractLesson

Tabla 19: Métodos de la clase AbstractLesson

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
getName()	Pública	Obtendrá el nombre del fichero jsp de la lección activa
setRanking(ranking)	Pública	Establece el orden en el que aparecerá la lección activa dentro de su categoría, en el menú de lecciones. Recibirá el parámetro Ranking, que será de tipo numérico entero.
isCompleted(sessionVar)	Pública	Indicará si la lección se ha superado o no. Recibirá el parámetro un objeto con las variables de

		sesión.
getCategory()	Pública	Devolverá el nombre de la categoría en la que se ha clasificado la lección.
getHint(sessionVar, hintNumber)	Pública	Devolverá una pista concreta de la lección. Recibirá como parámetro un objeto con las variables de sesión y el número de pista que se quiere obtener.
getLessonName()	Protegida	Devolverá el nombre de la lección Activa
getLessonPlan()	Pública	Devolverá un texto que contiene el plan a seguir para superar la lección
getSource(sessionVar)	Pública	Devolverá el código java de la lección en modo texto para que pueda ser mostrado por pantalla. Recibirá como parámetro un objeto con las variables de sesión
getSolution()	Pública	Devolverá un texto que informa de cómo solucionar la lección activa.

3.1.1.1.2 LessonAdapter

Tabla 20: Métodos de la clase LessonAdapter

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
createContent(sessionVar)	Protegida	Generará la interfaz de la lección, compuesta por código HTML y Javascript que será interpretado por el navegador. Como parámetro recibirá un objeto con las variables de sesión
getHints(sessionVar)	Protegida	Obtendrá la lista de las pistas de la lección activa. Recibirá como parámetro un objeto con las variables de sesión
getInstructions(sessionVar)	Pública	Obtendrá las instrucciones de la lección que se encuentran en el fichero del plan de la lección entre las etiquetas de comentario <!—Start Instructions - -> y <!—Stop Instructions - -> . Recibirá como parámetro un objeto con las variables de sesión
makeSuccess(sessionVar)	Protegida	Marcará la lección activa como superada. Como parámetro recibirá un objeto con las variables de sesión

3.1.1.1.3 Clickjacking

Tabla 21: Métodos de la clase Clickjacking

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
createContent(sessionVar)	Pública	Generará un contenedor con los siguientes elementos HTML y Javascript: ➢ Botón: contendrá el texto "Púlsame!!". Al recibir el clic, mostrará el siguiente mensaje: "Clic recibido por el botón"

		➢ Capa invisible: consistirá en un elemento *iframe* con las siguientes características: o Estará situado sobre el botón "Púlsame" o Tendrá opacidad 0 o El índice de profundidad será más alto que el del botón "Púlsame" ➢ Botón validar: recargará la lección para que los cambios de código que se hayan realizado surtan efecto ➢ Código Javascript: realizará todas las acciones de los elementos anteriormente descritos. Como parámetro recibirá un objeto con las variables de sesión
getDefaultCategory()	Pública	Devolverá un texto que indicará la categoría dentro de la que se ha clasificará la lección: "GENERAL"
getHints(sessionVar)	Pública	Devolverá la lista de pistas de la lección de Clickjacking. Como parámetro recibirá un objeto con las variables de sesión
getInstructions(sessionVar)	Pública	Devolverá el texto de las instrucciones de Clickjacking en el idioma que seleccionó el usuario. Como parámetro recibirá un objeto con las variables de sesión
getDefaultRanking()	Pública	Devolverá un número que indica el orden en que aparece la lección dentro de su sección en el menú de lecciones. El valor será 30
getTitle()	Pública	Devolverá un texto con el título de la lección: "Clickjacking"
getCredits()	Pública	Devolverá un texto con información relativa al autor de la lección.

3.1.1.1.4 EscaneoPuertosCSPP

Tabla 22: Métodos de la clase EscaneoPuertosCSPP

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
createContent(sessionVar)	Pública	Generará un contenedor con los siguientes elementos HTML y Javascript: ➢ Control de Acceso: consistirá en una tabla HTML que contenga tres elementos: o User: entrada tipo texto de nombre de usuario. o Password: entrada tipo texto de contraseña.

		○ Botón: realiza un envío de los campos “User” y “Password” al servidor. ➢ Botón validar: recargará la lección para que los cambios de código que se hayan realizado surtan efecto. Como parámetro recibirá un objeto con las variables de sesión
getDefaultCategory()	Pública	Devolverá un texto que indicará la categoría dentro de la que se ha clasificará la lección: “GENERAL”
getHints(sessionVar)	Pública	Devolverá la lista de pistas de la lección de Escaneo Anónimo de Puertos por CSPP. Como parámetro recibirá un objeto con las variables de sesión
getInstructions(sessionVar)	Pública	Devolverá el texto de las instrucciones de Escaneo Anónimo de Puertos por CSPP en el idioma que seleccionó el usuario. Como parámetro recibirá un objeto con las variables de sesión
getDefaultRanking()	Pública	Devolverá un número que indica el orden en que aparece la lección dentro de su sección en el menú de lecciones. El valor será 40
getTitle()	Pública	Devolverá un texto con el título de la lección: “Escaneo Anónimo de Puertos CSPP”
getCredits()	Pública	Devolverá un texto con información relativa al autor de la lección.

3.1.1.1.5 RepudiationAttack

Tabla 23: Métodos de la clase RepudiationAttack

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
createContent(sessionVar)	Pública	Generará un contenedor con los siguientes elementos HTML: ➢ Control de Acceso: consistirá en una tabla HTML formada por un conjunto de registros, compuestos a su vez por los campos: ○ Eliminación: contendrá un aspa para eliminar el registro. ○ Producto: Texto de ejemplo con el nombre de un producto. ○ Precio: Texto de ejemplo con el precio de un producto. Como parámetro recibirá un objeto con las variables de sesión
getDefaultCategory()	Pública	Devolverá un texto que indicará la categoría

		dentro de la que se ha clasificará la lección: "GENERAL"
getHints(sessionVar)	Pública	Devolverá la lista de pistas de la lección de Repudiation Attack. Como parámetro recibirá un objeto con las variables de sesión
getInstructions(sessionVar)	Pública	Devolverá el texto de las instrucciones de Repudiation Attack en el idioma que seleccionó el usuario. Como parámetro recibirá un objeto con las variables de sesión
getDefaultRanking()	Pública	Devolverá un número que indica el orden en que aparece la lección dentro de su sección en el menú de lecciones. El valor será 50
getTitle()	Pública	Devolverá un texto con el título de la lección: "Repudiation Attack"
getCredits()	Pública	Devolverá un texto con información relativa al autor de la lección.

3.1.2 Diseño de Componente: Gestor Multilenguaje

Se describirán aquellos elementos que harán posible visualizar las lecciones en distintos idiomas.

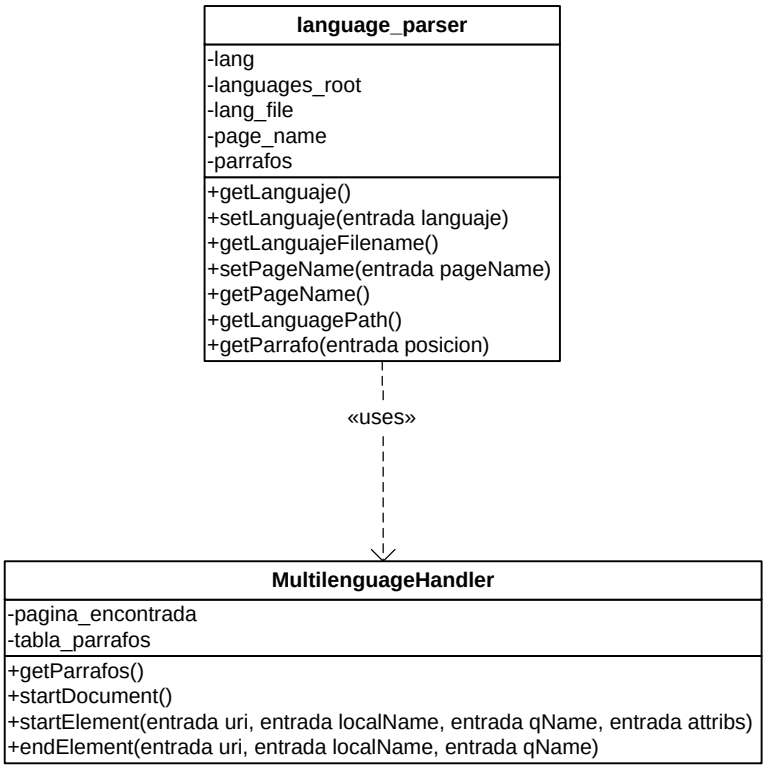

Ilustración 13 Diagrama de Clases del Componente Multilenguaje

Cada lección usará este componente para solicitar los textos explicativos en el idioma que el usuario haya seleccionado.

Al mismo tiempo este componente obtendrá los textos de una base de datos que se detallará más adelante.

Las dos clases que formarán el componente multilenguaje se exponen en la Tabla 24:

Tabla 24: Clases del componente Multilenguaje

NOMBRE	VISIBILIDAD	TIPO	DESCRIPCIÓN
Language_parser	Pública	Instanciable	Es la clase que instanciarán todas las lecciones para obtener sus textos
MultilanguageHandler	Pública	Instanciable	Es la clase que usará la clase Language_parser para obtener los textos de la base de datos XML. Será una extensión de la clase DefaultHandler, modificando y añadiendo nuevos métodos

Los métodos de cada una de las clases se detallan a continuación:

3.1.2.1.1 Language_parser:

Tabla 25: Métodos de la clase Language_parser

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
getLanguaje()	Pública	Devolverá el idioma en el que se mostrarán las lecciones
setLanguaje(languaje)	Pública	Establecerá el idioma en el que se mostrarán las lecciones. Recibirá como parámetro una cadena de texto con el código del idioma
getLenguajeFilename()	Pública	Obtendrá el prefijo del nombre de fichero del idioma en que se mostrarán las lecciones
setPageName(pageName)	Pública	Indicará el nombre de la página de la que se quieren obtener los textos. Recibirá como parámetro una cadena de texto con el nombre de la página
getLanguagePath()	Pública	Devolverá una cadena de texto con la ruta completa del fichero del idioma en el que se mostrará la lección
getParrafo(posición)	Pública	Devolverá el texto del párrafo que indicará el parámetro posición

3.1.2.1.2 MultilenguageHandler:

Tabla 26: Métodos de la clase MultilenguageHandler

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
getParrafos()	Pública	Obtendrá una tabla hash con todos los párrafos de una lección
startDocument()	Pública	Modificará el método original de DefaultHandler. Inicializará la variable que indica si se ha encontrado la lección a falso.
startElement(uri, localName, qName, attribs)	Pública	Modificará el método original de DefaultHnadler, Cargará la tabla hash con los textos de la lección.
endElement(uri, localName, qName)	Pública	Modificará el método original de DefaultHandler, vuelve a inicializar la variable que indica si se ha encontrado la lección a falso, pero solo en el caso de que se hayan encontrado los textos de la lección.

3.1.3 Diseño de Componente: Intérprete XML SAX

Este componente contendrá las clases necesarias para interpretar los ficheros XML de idioma utilizando la metodología SAX[7].

En la Ilustración 14 se describirán las clases del paquete SAX que se han utilizado para el desarrollo de este proyecto.

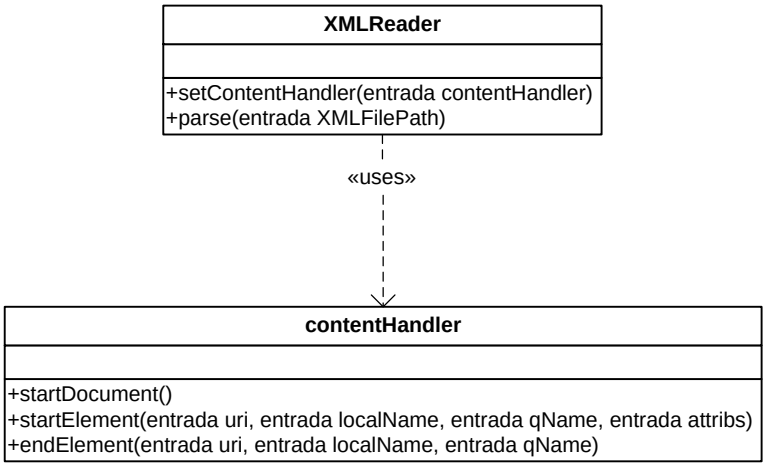

Ilustración 14 Diagrama de Clases del Intérprete XML SAX

[7] SAX: Interpreta los documentos XML de forma secuencial, hasta dar con el elemento buscado.

Las clases de la Ilustración 14 se detallarán en la Tabla 27:

Tabla 27: Clases del componente XML - SAX

NOMBRE	VISIBILIDAD	TIPO	DESCRIPCIÓN
XMLReader	Pública	Instanciable	Esta clase proporcionará la funcionalidad necesaria para leer los documentos XML de idiomas.
contentHandler	Pública	Instanciable	Definirá las reglas que permitan a la clase XMLReader interpretar los documentos XML

A continuación se explican los métodos de las clases incluidas en la Ilustración 14:

3.1.3.1.1 XMLReader

Tabla 28: Métodos de la clase XMLReader

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
setContentHandler(contentHandler)	Pública	Establece las directrices necesarias para interpretar el documento. contentHandler es el objeto que contiene las reglas para identificar los elementos XML del documento de idiomas.
Parse(XMLFilePath)	Pública	Obtiene los párrafos de la lección indicada del fichero cuya ruta ha sido pasada por el parámetro XMLFilePath

3.1.3.1.2 contentHandler

Tabla 29: Métodos de la clase contentHandler

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
startDocument()	Pública	El contenido de este método determinará las acciones a realizar cuando XMLReader comienza a leer el documento
startElement(uri, localName, qName, attribs)	Pública	Este método determinará las acciones a realizar cuando XMLReader comienza a leer un elemento del documento. El nombre del elemento que se va a comenzar a leer vendrá determinado por el parámetro qName
endElement(uri, localName, qName)	Pública	Este método determinará las acciones a realizar cuando XMLReader termina de leer el documento. El nombre del elemento que se acaba de leer vendrá determinado por el parámetro qName

3.1.4 Diseño de Componente: Base de Datos XML de Idiomas

Consistirá en un conjunto de ficheros de texto XML que todos los textos de las lecciones en los distintos idiomas.

Cada fichero de texto contendrá todos los párrafos del sistema en un idioma diferente.

Todos los ficheros de idioma tienen que seguir una misma estructura con el fin de que puedan ser interpretados por el lector XML explicado en la sección anterior.

Esta estructura viene determinada por otro fichero de texto llamado "*Documento de Tipo de Datos"* o DTD, que contiene lo indicado en la Ilustración 15:

```
<!ELEMENT parseador pagina+>
<!ELEMENT pagina parrafo+>
<!ATTLIST pagina nombre CDATA #REQUIRED>
<!ELEMENT parrafo ANY>
<!ATTLIST parrafo posicion CDATA #REQUIRED>
<!ATTLIST parrafo texto CDATA #REQUIRED>
```

Ilustración 15: DTD Multilenguaje

Para una mejor comprensión, las explicaciones de este código se detallarán a continuación:

1. Cada fichero (llamado "*parseador*") de idioma, estará formado por al menos un elemento llamado "*página*".
2. Cada elemento "*página*" estará formado por al menos un elemento "*párrafo*" que contendrá el texto que leerá el usuario final de la aplicación.
3. Cada elemento "*párrafo*" tendrá dos atributos:
    a. Atributo "*posición*": está formado por un identificador único por cada página, y determinará el lugar donde se situará el texto dentro de la plantilla HTML de la lección. Este identificador estará situado en estas plantillas para ser sustituidos por los textos, y es por esta razón por la que debe contener un formato único que no se confunda con ninguna otra cadena de texto que puedan existir en la plantilla. Para lograrlo, se ha optado por utilizar este formato: *_#[posición]#_*, donde:
        i. "*_#*" y "*#_*": indican al parseador[8] de idiomas que entre estos dos conjuntos de caracteres se encuentra el índice de posición
        ii. [posición]: contendrá un número que será único por cada elemento "*página*".
    b. Atributo "*texto*": contendrá el texto que se situará en el lugar indicado por el atributo "*posición*".

[8] Mecanismo que sitúa los textos en el lugar adecuado dentro de las plantillas

3.1.5 Diseño de Componente: Servlets

Este componente realizará la gestión de comunicaciones del servidor con el equipo cliente.

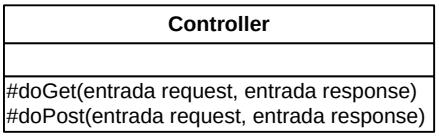

Ilustración 16 Diagrama de Clases del Servlet

El controlador está formado por una sola clase que se detallará en la Tabla 30:

Tabla 30: Clases del componente Servlets

NOMBRE	VISIBILIDAD	TIPO	DESCRIPCIÓN
Controller	Pública	Instanciable	Esta clase recibirá las peticiones del cliente y devuelve la respuesta generada por el servidor.

Los métodos de la Ilustración 16 se expondrán a continuación:

3.1.5.1.1 Controller

Tabla 31: Métodos de la clase Controller

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
doGet(request, response)	Protegida	Esté método llamará a la función doPost() descrito a continuación, de forma que todas las peticiones (por Get o por Post) se tratarán de la misma forma.
doPost(request, response)	Protegida	Recibirá la petición del cliente a través del parámetro request y enviará la respuesta indicada por el parámetro response.

3.1.6 Diseño de Componente: Sesión de Usuarios

Este componente está formado por un conjunto de clases encargadas de gestionar los datos relativos a las sesiones de usuario.

WebGoat cuenta con numerosas clases relacionadas con el control de sesiones, pero para este proyecto únicamente expondremos tres de ellas, que son las más representativas y cuyo estudio ha resultado útil para el desarrollo de las nuevas lecciones a implementar y para la implementación de la solución para el multilenguaje.

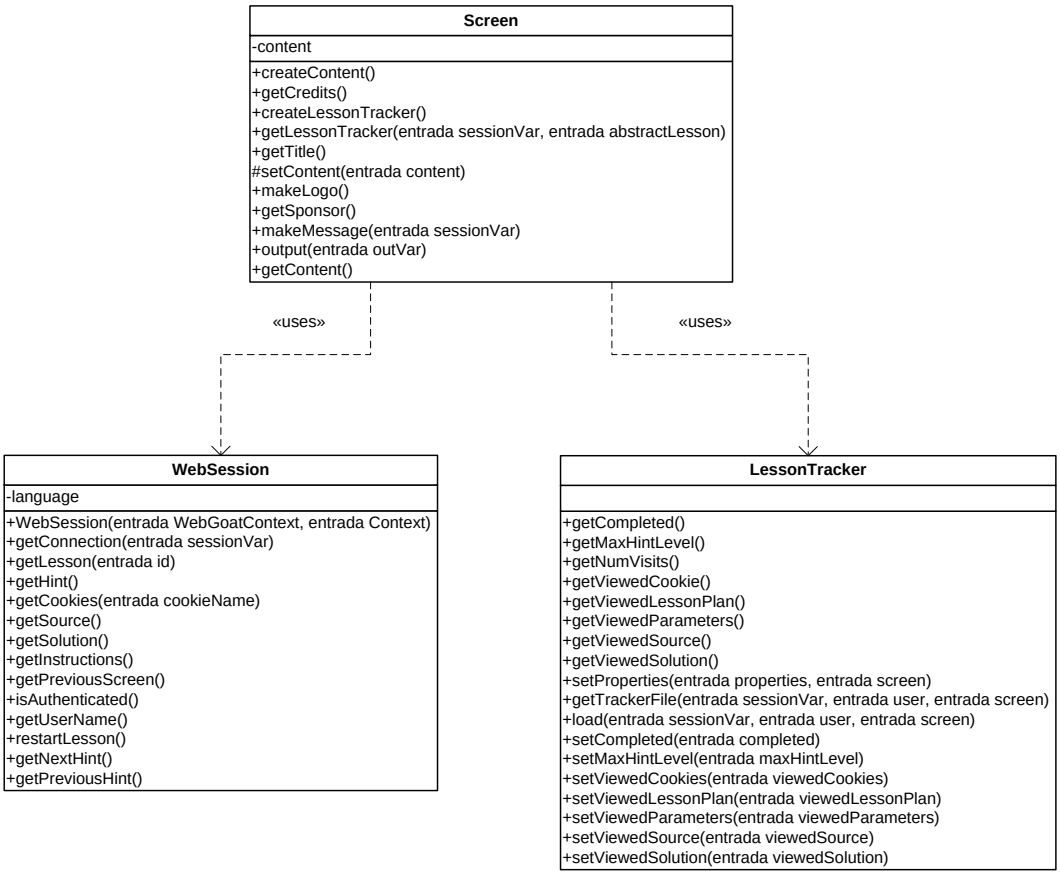

Ilustración 17 Diagrama de Clases de Sesión de Usuario

Las clases de la Ilustración 17 se explicarán en la Tabla 32:

Tabla 32: Clases del componente de Sesión de Usuarios

NOMBRE	VISIBILIDAD	TIPO	DESCRIPCIÓN
Screen	Pública	Instanciable	Esta clase actuará como la interfaz entre las clases propias de control de sesión y el contenido de las lecciones.
WebSession	Pública	Instanciable	Gestionará variables de sesión relacionadas con las lecciones que visita el usuario. Desde esta clase se controlará en qué idioma se muestran las lecciones, ya que contiene el atributo "*language*" que indicará el código de lenguaje seleccionado por el usuario.
LessonTracker	Pública	Instanciable	Registrará todas las acciones que se realizan con la lección.

Los métodos de las clases descritas en el cuadro anterior se expondrán a continuación:

3.1.6.1.1 Screen

Tabla 33: Métodos de la clase Screen

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
createContent()	Pública	Método abstracto que será redefinido por la clase LessonAdapter para generar el contenido de la lección
getCredits()	Pública	Método abstracto que será redefinido por la clase que define las propiedades y métodos propios de cada lección, y contendrá un texto informativo con el nombre de los autores de la lección
creteLessonTracker()	Pública	Creará una instancia de la clase LessonTracker asociada a la lección activa.
getLessonTracker(sessionVar, abstractLesson)	Pública	Obtendrá una instancia de LessonTracker dada una variable de sesión y una lección dada por la variable abstractLesson
getTitle()	Pública	Método abstracto que será redefinido por la clase que define las propiedades y métodos propios de cada lección, y contendrá el título de la lección.
setContent(content)	Protegida	Establecerá el contenido de la lección
makeLogo()	Pública	Devolverá un elemento *anchor*[9] HTML para obtener un logotipo. Por defecto obtiene el de una empresa patrocinadora: http://www.aspectsecurity.com/webgoat.html
getSponsor()	Pública	Devolverá una cadena de texto con el nombre del patrocinador, por defecto será: "Aspect Security"
makeMessage(sessionVar)	Pública	Devolverá un mensaje de texto indicado por la variable sessionVar
Output(outVar)	Pública	Imprime por pantalla el contenido de la lección
getContent()	Pública	Devolverá el contenido de la lección

[9] Elemento HTML que sirve para crear enlaces de tipo Hperlink, de forma que al pulsar sobre el mismo el usuario será redirigido a la dirección indicada por el atributo "href" del elemento

3.1.6.1.2 WebSession

Tabla 34: Métodos de la clase WebSession

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
webSession(webgoatContext, Context)	Pública	Actuará como método constructor de la clase, inicializando los atributos a partir de los parámetros webgoatContext y Context.
getConnection(sessionVar)	Pública	Devolverá la conexión a la base de datos a partir de los datos indicados por el parámetro sessionVar
getLesson(id)	Pública	Devolverá un objeto del tipo abstractLesson asociado a la lección indicada por el parámetro id.
getHint()	Pública	Obtendrá una pista de la lección
getCookies(cookieName)	Pública	Obtendrá el contenido de la cookie cuyo nombre será el obtenido del parámetro cookieName.
getSource()	Pública	Devolverá el código fuente java de la lección activa. Si esta función no es redefinida en las clases que la extienden, devolverá un mensaje de error indicando que la función no está disponible
getSolution()	Pública	Devolverá el texto con la solución de la lección activa. Si esta función no es redefinida en las clases que la extienden, devolverá un mensaje de error indicando que la función no está disponible
getInstructions()	Pública	Devolverá el texto con las instrucciones de la lección activa.
getPreviousScreen()	Pública	Obtendrá la pantalla anterior realizando la misma función que realizan los navegadores con la opción “Atrás”
isAuthenticated()	Pública	Informará si el usuario está autenticado o no en el sistema.
getUserName()	Pública	Devolverá el nombre del usuario activo en el sistema.
restartLesson()	Pública	Reinicia la lección, de forma que la próxima vez que el usuario la visite será como si la visitase por primera vez.
getNextHint()	Pública	Devolverá la siguiente pista a mostrar en la lección activa.
getPreviousHint()	Pública	Obtendrá la pista anterior a la que se está mostrando en la lección activa.

3.1.6.1.3 LessonTracker

Tabla 35: Métodos de la clase LessonTracker

NOMBRE	VISIBILIDAD	DESCRIPCIÓN
getCompleted()	Pública	Indicará si la lección ha sido superada o no.
getMaxHintLevel()	Pública	Devolverá el número de pistas que ha visto el usuario en la lección activa.
getNumVisits()	Pública	Obtendrá el número de veces que un usuario realiza a la lección activa
getViewedCookie()	Pública	Indicará si el usuario ha visto las cookies en la lección activa
getViewedLessonPlan()	Pública	Indicará si el usuario ha visto el plan de la lección activa.
getViewedParameters()	Pública	Indicará si el usuario ha visto o no los parámetros de la lección activa
getViewedSource()	Pública	Indicará si el usuario ha visto el código fuente de la lección activa
getViewdSolution()	Pública	Indicará si el usuario ha visto la solución de la lección activa
setProperties(properties, screen)	Pública	Establece un conjunto de propiedades indicadas por el parámetro "*properties*" en la lección indicada por el parámetro screen.
getTrackerFile(sessionVar, user, screen)	Pública	Devolverá el fichero de registro de actividad a partir de los parámetros de sesión, usuario y screen.
Load(sessionVar,user,screen)	Pública	Cargará la información de registro de actividad para la lección indicada por el parámetro screen, user y sessionVar.
setCompleted(completed)	Pública	Establecerá la lección activa como completada o no completada, según indique el parámetro completed.
setMaxHintLevel(maxHintLevel)	Pública	Establecerá el número máximo de pistas vistas por el usuario para la lección activa
setViewedCookies(viewedCookies)	Pública	Establecerá si se han visto las cookies o no, según indique el parámetro viewdCookies.
setViewedLessonPlan(viewedLessonPlan)	Pública	Indicará al sistema si el usuario ha visto o no el plan de la lección activa, según indique el parámetro viewdLessonPlan
setViewedParameters(viewedParameters)	Pública	Indicará al sistema si el usuario ha

		visto los parámetros de la lección o no, según indique el parámetro viewedParameters.
setViewedSource(viewedSource)	Pública	Indicará si el usuario ha visto el código java de la lección activa, según indique el parámetro viewedSource.
setViewedSolution(viewedSolution)	Pública	Indicará si el usuario ha visto la solución de la lección activa, según indique el parámetro viewedSolution.

3.1.7 Diseño de Componente: Base de Datos HSQLDB

Este componente contiene estará formado por dos clases que gestionarán la base de datos y por una base de datos HSQLDB que serán usadas por algunas lecciones.

Para la elaboración de este proyecto no ha sido necesario utilizar este componente, por lo que en este documento simplemente se informará de la existencia del mismo y de la estructura de sus clases.

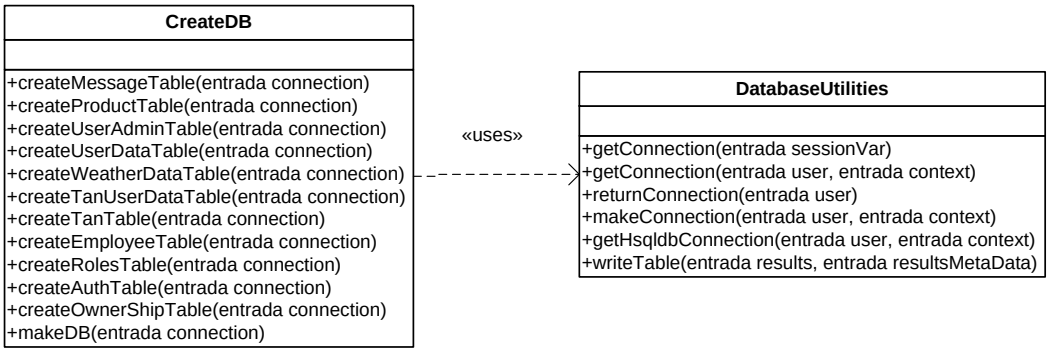

Ilustración 18 Diagrama de Clases de Base de Datos HSQLDB

3.1.8 Diseño de Componente: Utilidades

Este componente está formado por clases que ofrecen funcionalidades extra que se pueden necesitar en cualquier lección.

En las nuevas lecciones no será necesario el uso de ninguna de estas funcionalidades, por lo que este documento contendrá un diagrama de las clases que forman el componente sin entrar a detallar el contenido de las mismas:

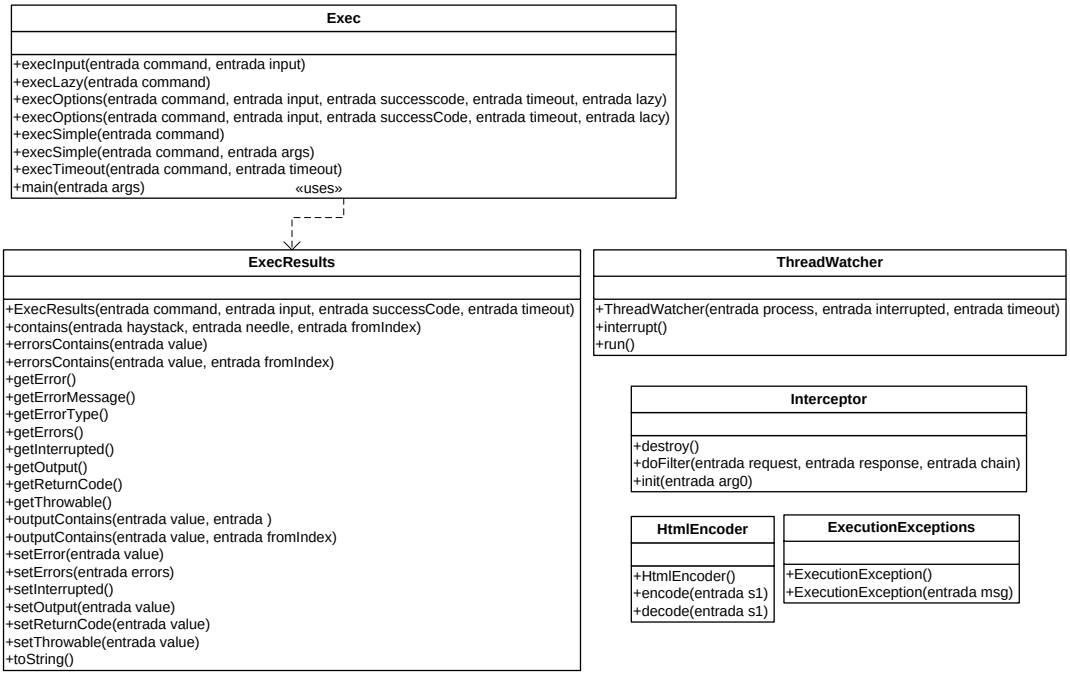

Ilustración 19 Diagrama de Clases de Utilidades

3.1.9 Diseño de Componente: Contenidos

Este componente contendrá una estructura de directorios y ficheros que contienen plantillas, imágenes, códigos Javascript y textos con los contenidos de las lecciones que forman parte de WebGoat.

3.2 Diagramas de Secuencia

Partiendo de los casos de uso expuestos en el apartado "*Diagrama de Casos de Uso*" dentro de la sección de Análisis, se representarán los diagramas de secuencia asociados a cada caso de uso.

De esta forma especificaremos la solución que se va a implementar para cada uno de los casos.

La estructura de esta sección se realizará en base a los casos de uso tal y como se expone a continuación.

3.2.1 Cambiar Idioma

Este diagrama de secuencia detallará la solución aportada para que un usuario pueda seleccionar el idioma en el que desee ver las lecciones:

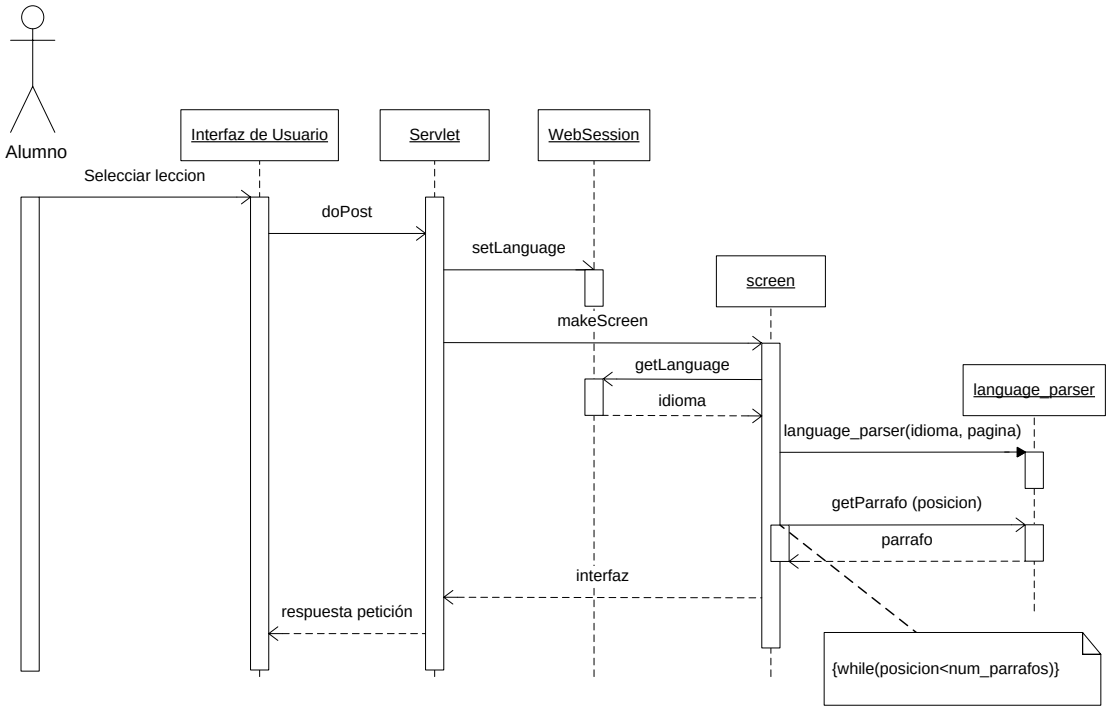

Ilustración 20 Diagrama de Secuencia - Cambio de Idioma

Para seleccionar un idioma el alumno deberá registrarse en el sistema, y una vez se le muestre la pantalla de bienvenida, seleccionar en el formulario contenido en esta interfaz, el idioma en el que necesite visualizar los contenidos de la aplicación.

Al realizar la selección en la interfaz, se envía una petición al servidor, que es recibida por el servlet. Este objeto, indicará al objeto WebSession (encargado de gestionar las variables de sesión) el idioma en el que deberán cargarse los contenidos del sistema.

Seguidamente, solicita al objeto screen que genere una pantalla o interfaz de respuesta para enviar al usuario.

El objeto screen, solicitará al objeto WebSession, el idioma en el que se quieren obtener los textos a incluir en la interfaz de respuesta.

Con el idioma obtenido, el objeto screen solicitará los textos de la página de bienvenida en el idioma indicado sustituyéndolos por la etiqueta correspondiente en la plantilla de la página de bienvenida.

Finalmente, la interfaz en el nuevo idioma se enviará al servlet, que a su vez lo devuelve al navegador del usuario.

3.2.2 Examinar Lección

La solución aportada a continuación resuelve el caso de uso que consiste en la forma en la que se le van a enviar los datos al usuario para que este pueda visualizar y aprender el contenido de las lecciones.

Como se refleja en el apartado de Casos de Uso, nos encontraremos ante un escenario básico y seis escenarios alternativos, cuyos diagramas de secuencia y sus explicaciones asociadas se detallarán a continuación.

Los diagramas que se mostrarán en esta sección incorporarán dos objetos que no han sido tratados en apartados anteriores, ya que se expresan de este modo para facilitar la legibilidad:

- Interfaz de Usuario: Se corresponderá con una ventana del navegador de Internet que esté utilizando el usuario para acceder a la aplicación.
- Lesson: Se trata de un objeto genérico que representa a cada uno de los objetos de las lecciones que forman parte de WebGoat.

3.2.2.1 Escenario Básico de Examinar Lección: Cargar Contenido de Lección

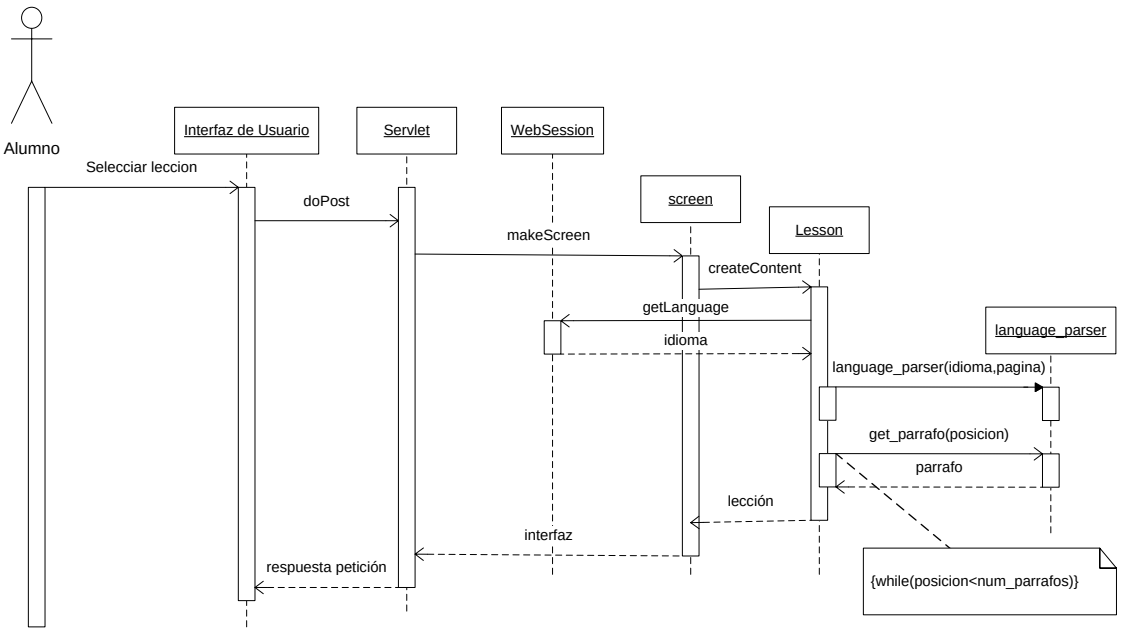

Ilustración 21 Diagrama de Secuencia - Carga de Contenido de Lección

El usuario seleccionará una lección en el menú de la interfaz y este enviará una petición al servidor que será recogida por el servlet.

El servlet analizará la petición y creará un objeto screen para generar la respuesta con la lección solicitada.

El objeto screen generará el contenido de la lección a través de Lesson, que sería el objeto asociado a la lección solicitada por el usuario.

Lesson solicitará el idioma al objeto WebSession, y los textos al objeto language_parser, para generar la lección solicitada.

Lesson, devolverá la lección al objeto screen que a su vez se lo enviará al objeto servlet, que enviará la respuesta a la interfaz de usuario para que el alumno pueda visualizarlo.

3.2.2.1.1 Escenario Alternativo 1 de Examinar Lección: Ver Lesson Plan

El contenido de Lesson Plan se cargará con el contenido de la lección, por tanto cuando el usuario lo solicite, la información estará cargada en su explorador, de forma que el usuario no podrá verla hasta que seleccione la opción “Lesson Plan” en el menú superior de la lección.

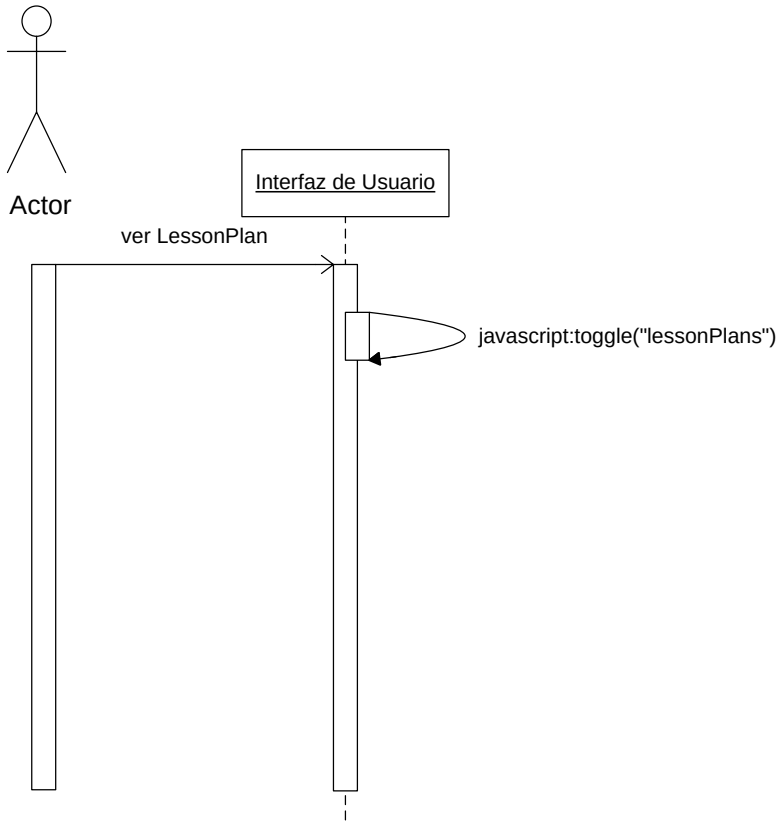

Ilustración 22 Diagrama de Secuencia - Ver Lesson Plan

Al pulsar sobre la opción “Lesson Plan”, se le mostrará al usuario el texto oculto con el contenido del plan de la lección.

3.2.2.1.2 Escenario Alternativo 2 de Examinar Lección: Ver Solución de la Lección

El contenido de este escenario no se cargará con la información de la lección, por tanto esta información deberá ser solicitada de forma específica al servidor cada vez que el usuario lo requiera.

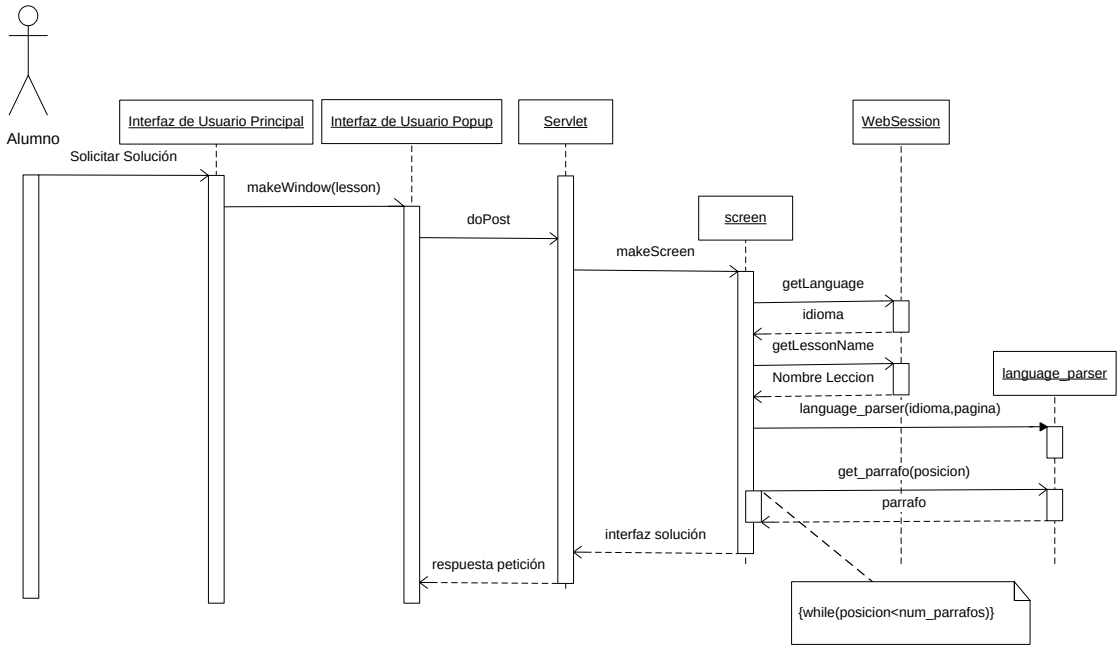

Ilustración 23 Diagrama de Secuencia - Ver Solución de Lección

El usuario seleccionará la opción “Lesson Solution” desde el menú superior de una lección.

Se abrirá otra ventana del navegador que solicitará al servidor que le envíe la solución de la lección activa.

El servlet recogerá la petición, y solicitará la interfaz de respuesta al objeto screen.

El objeto screen obtendrá del objeto WebSession, el nombre de la lección activa y el idioma en el que está navegando el usuario, para solicitar los textos de la solución al objeto language_parser y sustituirlos en la plantilla de la solución asociada a la lección activa.

Una vez construido, el mensaje de respuesta será enviado como respuesta al objeto servlet, que a su vez lo enviará a la ventana popup de navegador que se le habrá abierto al usuario.

3.2.2.1.3 Escenario Alternativo 3 de Examinar Lección: Ver Pista

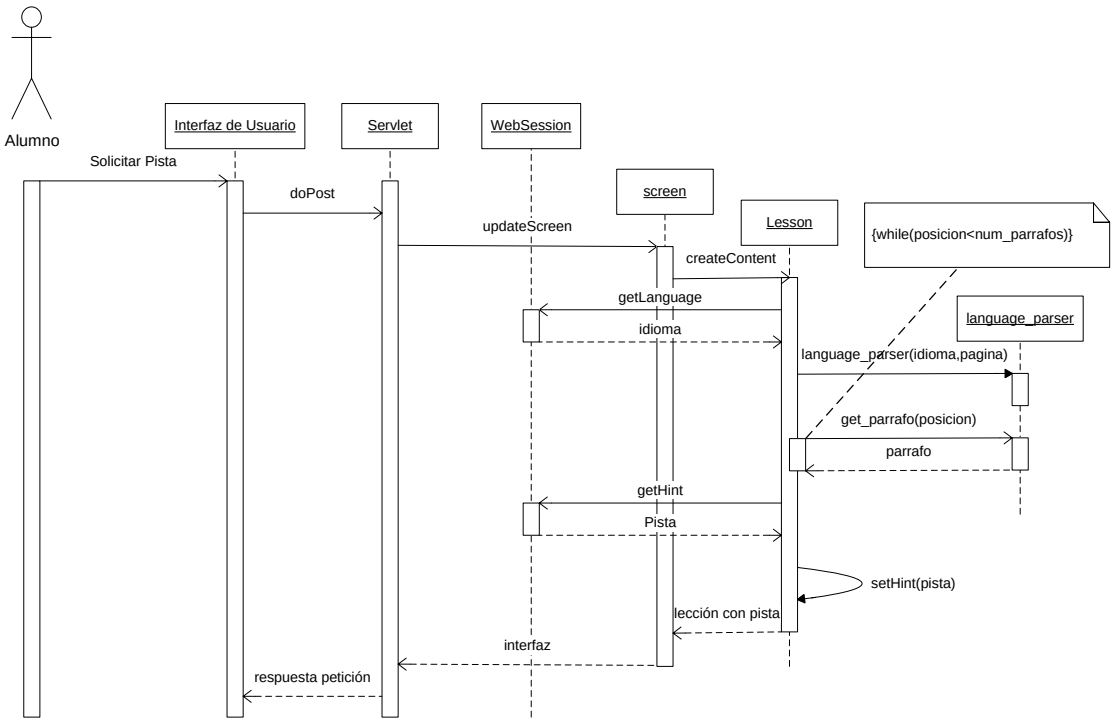

Ilustración 24 Diagrama de Secuencia - Ver Pista

Al pulsar sobre la opción “Next Hint” o “Previous Hint”, la interfaz envía una petición que será recogido por el servlet para solicitar una interfaz con la lección activa y la pista siguiente o la anterior.

La petición del objeto servlet es atendida por el objeto screen, que solicitará al objeto *lesson* que genere el código con el contenido deseado.

El objeto Lesson, solicitará el idioma del usuario, los textos asociados a la lección e idioma activos e incluirá la pista solicitada en el contenido.

El código de la lección con el contenido se devolverá al objeto screen, que a su vez lo enviará al objeto servlet para que responda a la petición inicial del usuario en la interfaz.

3.2.2.1.4 Escenario Alternativo 4 de Examinar Lección: Ver Código Java

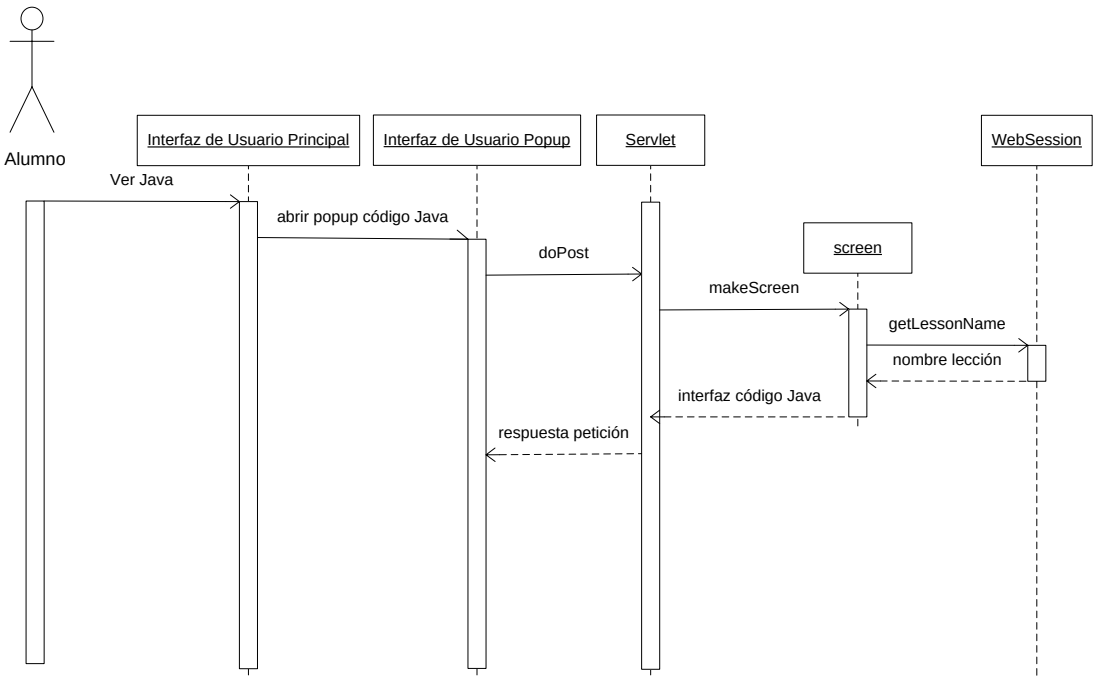

Ilustración 25 Diagrama de Secuencia - Ver Código Java

Al seleccionar la opción "Show Java" desde el menú superior de una lección se abrirá otra ventana del navegador que solicitará al servidor que le envíe el código Java de la lección activa.

El servlet recogerá la petición, y solicitará la interfaz de respuesta al objeto screen.

El objeto screen obtendrá del objeto WebSession, el nombre de la lección activa por la que esté navegando el usuario, para obtener el fichero de código Java correspondiente a esta lección.

Una vez construido, el mensaje de respuesta será enviado como respuesta al objeto servlet, que a su vez lo enviará a la ventana popup de navegador que se le habrá abierto al usuario.

3.2.2.1.5 Escenario Alternativo 5 de Examinar Lección: Ver Cookies

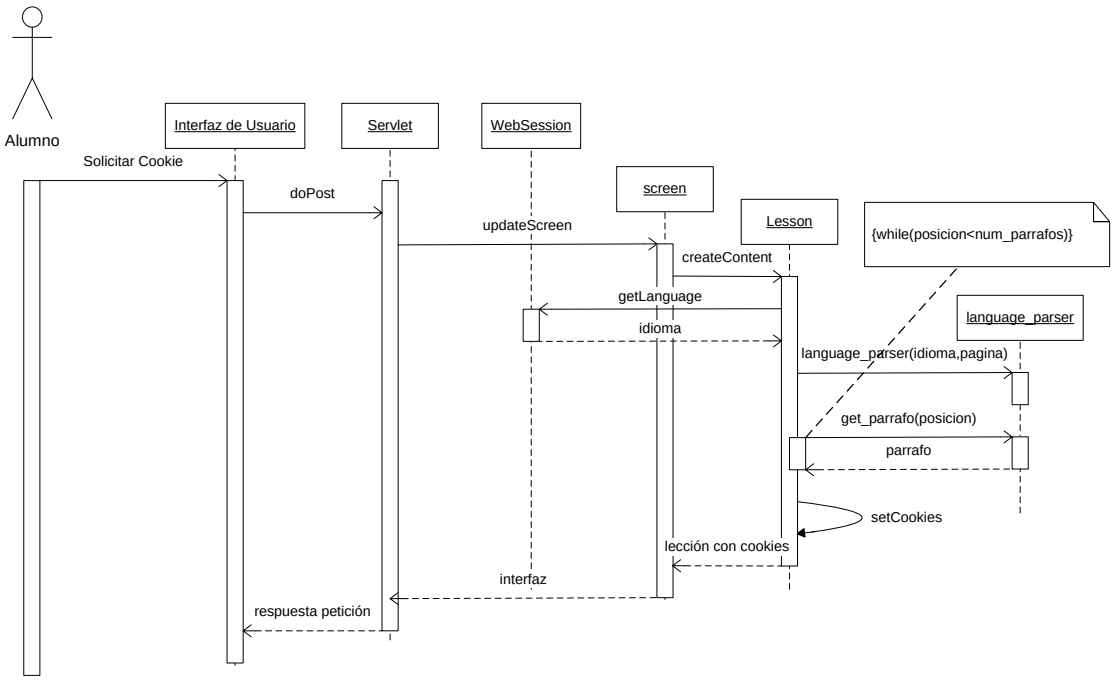

Ilustración 26 Diagrama de Secuencia - Ver Cookies

El usuario seleccionará la opción “Show Cookies” desde el menú superior de la lección.

La interfaz envía una petición que será recogido por el servlet para solicitar una interfaz con la lección activa y las cookies correspondientes.

La petición del objeto servlet será atendida por el objeto screen, que solicitará al objeto lesson que genere el código con el contenido deseado.

El objeto Lesson, solicitará el idioma del usuario, los textos asociados a la lección e idioma activos e incluirá las cookies en el contenido.

El código de la lección con el contenido se devolverá al objeto screen, que a su vez lo enviará al objeto servlet para que responda a la petición inicial del usuario en la interfaz.

3.2.2.1.6 Escenario Alternativo 6 de Examinar Lección: Ver Parámetros

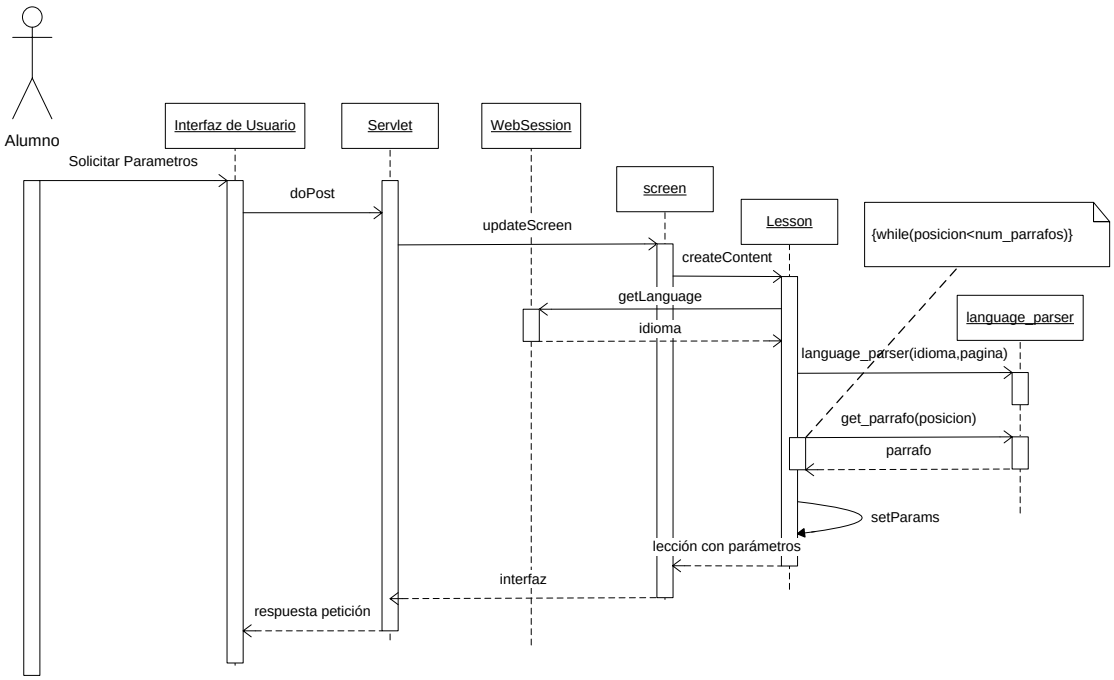

Ilustración 27 Diagrama de Secuencia - Ver Parámetros

Al pulsar sobre la opción “Show Params”, la interfaz envía una petición que será recogido por el servlet para solicitar una interfaz con la lección activa y los parámetros que son enviados por GET y por POST.

La petición del objeto servlet será atendida por el objeto screen, que solicitará al objeto lesson que genere el código con el contenido deseado.

El objeto Lesson, solicitará el idioma del usuario, los textos asociados a la lección e idioma activos e incluirá los parámetros obtenidos en el contenido.

El código de la lección con el contenido se devolverá al objeto screen, que a su vez lo enviará al objeto servlet para que responda a la petición inicial del usuario en la interfaz.

3.2.3 Superar Lección

Los diagramas de secuencia mostrados a continuación reflejan la solución aportada para que los alumnos superen las nuevas lecciones implementadas en el sistema.

Se ha escogido como escenario básico la lección de Clickjacking y como escenarios alternativos las lecciones de Escaneo Anónimo de Puertos por CSPP y Repudiation Attack.

3.2.3.1 Escenario Básico Superar Lección: Superar Clickjacking

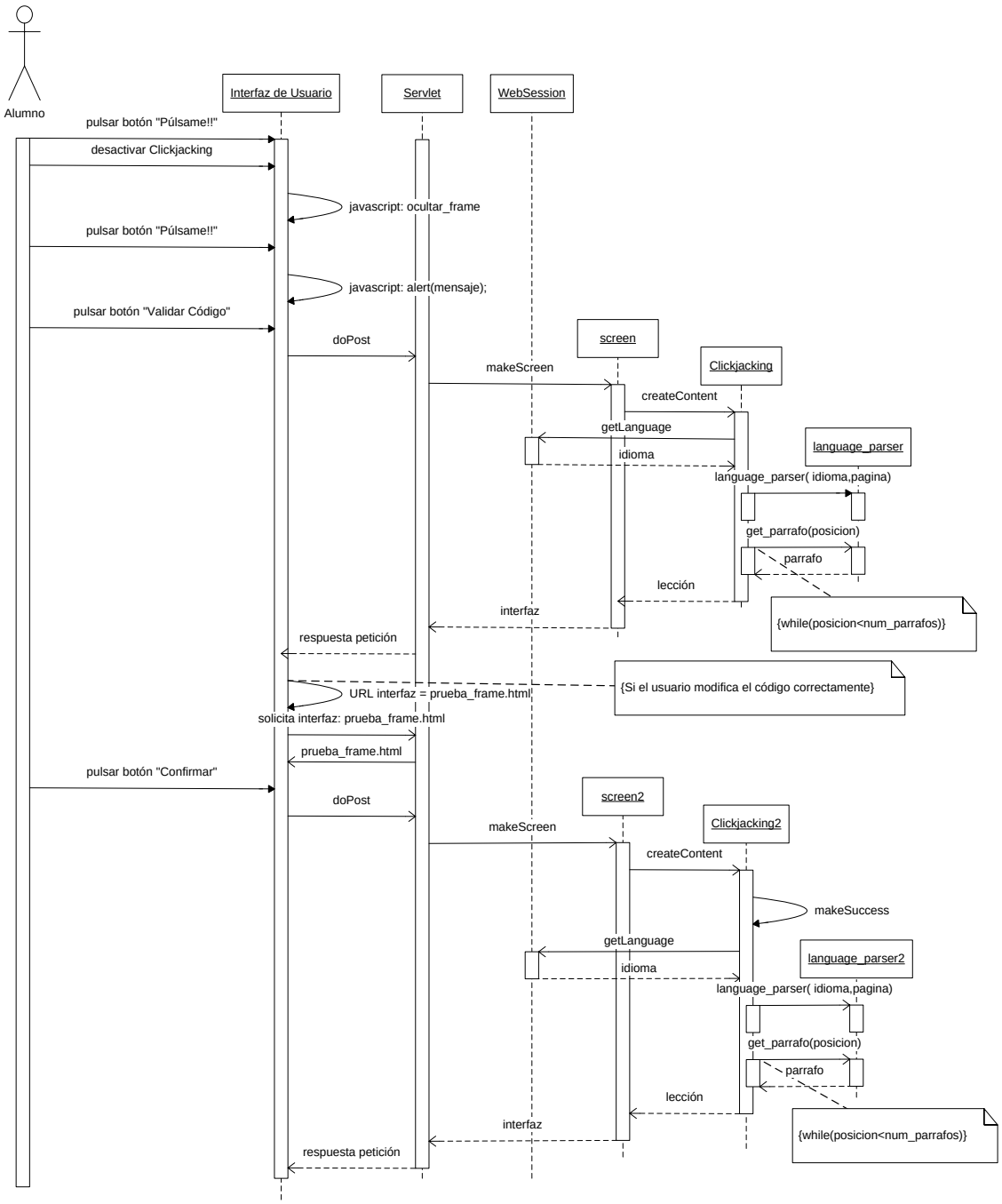

Ilustración 28 Diagrama de Secuencia - Superar Lección Clickjacking

Una vez cargada la lección de Clickjacking, el alumno se encontrará ante una simulación de Clickjacking que podrá activar y desactivar para comprobar el efecto del ataque.

El alumno intentará pulsar el botón “Púlsame!!” para observar el comportamiento.

Después indicará en el formulario que se desactive el Clickjacking, provocando que el *iframe* situado sobre el botón pase a estar debajo de este.

El alumno volverá a intentar pulsar el botón “Púlsame!!”, y al encontrarse este sobre el *iframe*, sí recibirá el clic.

Al recibir la pulsación, el botón “Púlsame!!” mostrará un mensaje al usuario indicando que ha recibido el clic.

Una vez el alumno haya comprendido y asimilado la funcionalidad de Clickjacking podrá modificar el código fuente del *frame* no visible, para que cada vez que se cargue, indique al navegador por Javascript, que todo elemento que contenga el frame, tenga la misma URL que el propio frame, de esta forma siempre se mostrará el contenido del mismo aunque este lo sitúen dentro de elementos no visibles.

Una vez modificado el código, el alumno podrá comprobar si ha superado la lección pulsando el botón “validar código”.

El navegador enviará una solicitud al servidor para recargar la lección, el servlet recibirá la petición y solicitará al objeto screen la interfaz de respuesta. El objeto screen solicitará los contenidos de la respuesta del objeto Clickjacking y este obtendrá el idioma del usuario del objeto WebSession y los textos del objeto language_parser para crear la interfaz de respuesta. El objeto screen devolverá la interfaz al servlet que a su vez lo enviará a la interfaz de usuario.

Para superar la lección, el alumno deberá modificar el código Javascript de la página cargada dentro del *iframe*, de forma que se asigne a la URL del navegador, la dirección de esta página.

Si el alumno logra superar la lección, al cargarla de nuevo, el código Javascript modificado hará que la página que se muestre en el navegador sea la cargada dentro del *iframe*.

El nuevo contenido mostrado al alumno contendrá un botón llamado “Confirmar” que el alumno deberá pulsar para superar la lección.

Al pulsar este botón, se volverá a llamar a la lección de Clickjacking indicando que ha sido superada.

El servlet solicitará al objeto screen la interfaz de respuesta con la lección Clickjacking superada.

El objeto screen a su vez solicitará los contenidos de la lección al objeto Clickjacking.

El objeto Clickjacking obtendrá el idioma de objeto WebSession y los textos del objeto language_parser. Además incluirá un texto indicando que la lección ha sido superada, y actualizará el estado de la lección en el sistema, marcándola como lección superada.

El objeto Clickjacking devolverá al objeto screen el contenido de la lección, y el objeto screen se lo enviará al servlet para que este se lo envíe a la interfaz de usuario.

3.2.3.2 Escenario Alternativo 1 Superar Lección: Superar Escaneo Anónimo de Puertos por CSPP

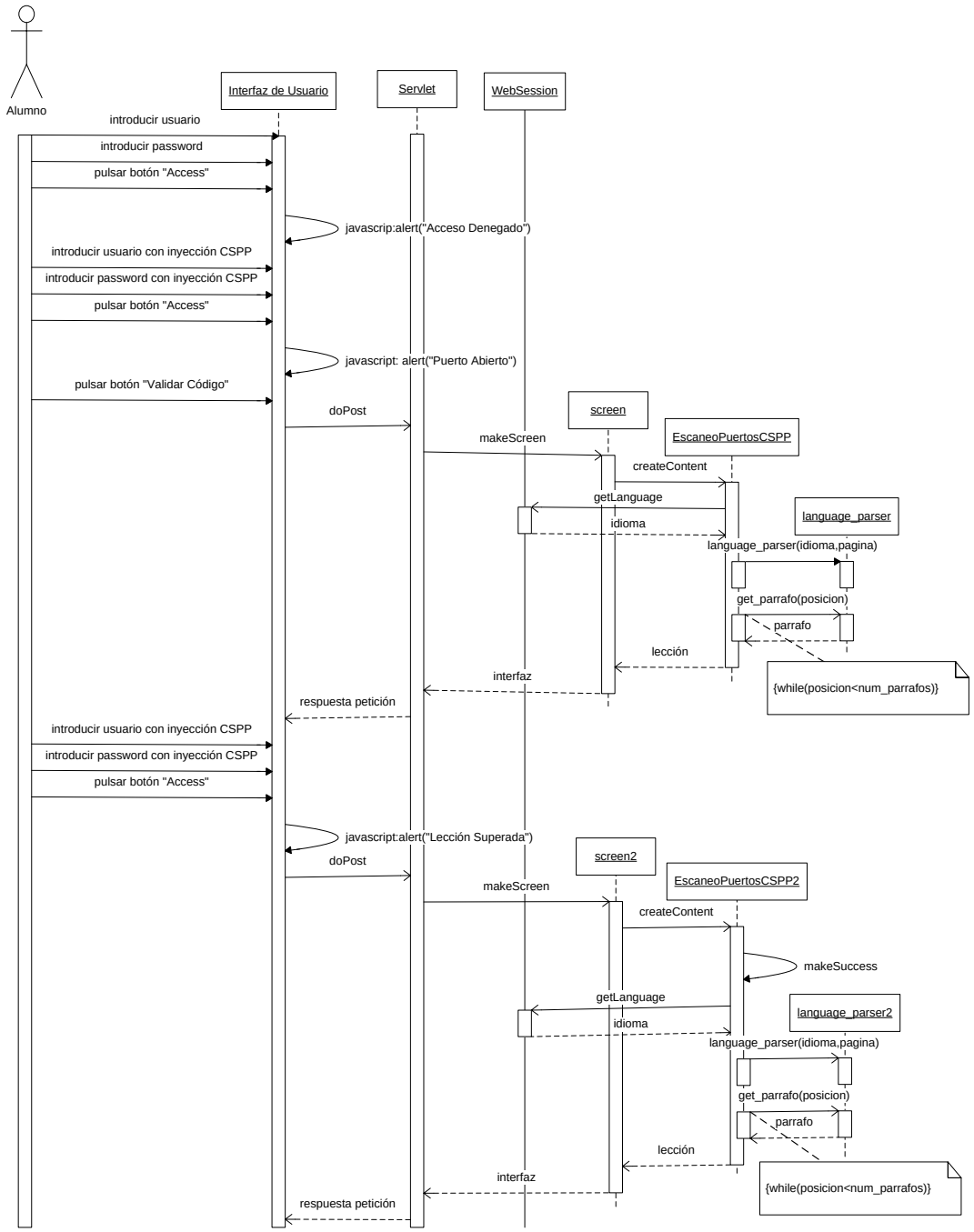

Ilustración 29 Diagrama de Secuencia - Superar Lección Escaneo Anónimo de Puertos CSPP

El alumno introducirá un nombre de usuario y una contraseña en el formulario de simulación de acceso a una base de datos y pulsará el botón “ACCESS” para simular el envío de credenciales.

Si el usuario no introduce una inyección CSPP, se le mostrará un mensaje informativo con el texto “Acceso Denegado”.

A continuación, el alumno introducirá una inyección CSPP para escaneo anónimo de puertos, utilizando los campos de usuario y password. Al pulsar el botón “ACCESS” se le mostrará un mensaje indicando que el puerto indicado en la inyección está abierto.

El alumno deberá modificar el código fuente del fichero “WebContent/lessons/EscaneoPuertosCSPP/antiCSPP.js” para evitar las inyecciones CSPP.

Esta modificación consiste en introducir en el fichero indicado un algoritmo en Javascript que detecte que se ha introducido una inyección del tipo CSPP.

Para comprobar si las modificaciones han tenido éxito, el alumno pulsará sobre el botón “Validar Código” para recargar la lección.

La interfaz solicitará la lección de Escaneo Anónimo de puertos por CSPP. La petición será recogida por el servlet que a su vez solicitará al objeto screen una interfaz para enviar como respuesta.

El objeto screen obtendrá la interfaz solicitando el contenido al objeto EscaneoPuertosCSPP, que obtendrá el idioma del usuario mediante el objeto WebSession y los textos del objeto language_parser.

El objeto EscaneoPuertosCSPP enviará el contenido de la lección al objeto screen que a su vez enviará la interfaz creado al servlet que lo devolverá a la interfaz de usuario.

Para comprobar el éxito de las modificaciones realizadas, el alumno deberá volver a realizar una inyección de Escaneo Anónimo de Puertos por CSPP. Al pulsar el botón “ACCESS” habiendo realizado la inyección CSPP, el sistema detectará que se ha intentado realizar una inyección y que ha sido evitada por las modificaciones realizadas por el alumno.

Si las modificaciones han tenido éxito se le mostrará un mensaje al alumno con el texto “Lección Superada”. Además se enviará una petición al servidor para cargar la lección de Escaneo Anónimo de Puertos por CSPP como lección superada.

3.2.3.3 Escenario Alternativo 2 Superar Lección: Superar Repudiation Attack

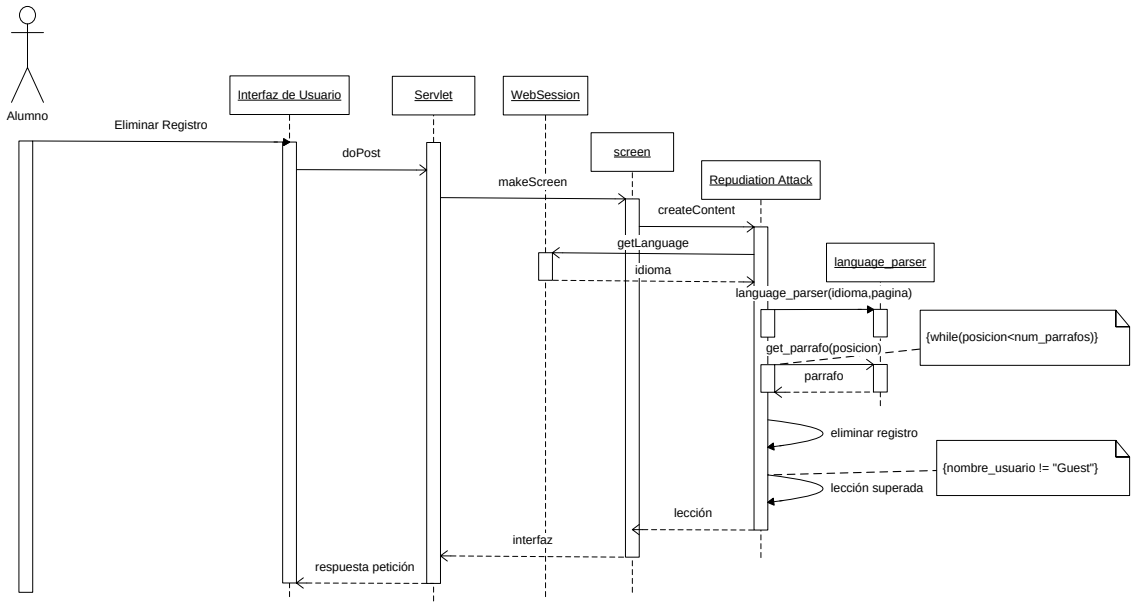

Ilustración 30 Diagrama de Secuencia - Superar Lección Repudiation Attack

El alumno eliminará un registro de la tabla que muestra la interfaz.

La interfaz enviará una solicitud de eliminación al servidor mediante una solicitud de recarga de la lección, indicando el nombre del usuario “Guest” y el registro a eliminar.

La solicitud será recogida por el servlet, que realizará una petición al objeto screen para obtener la interfaz de respuesta.

El objeto screen solicitará el contenido de la interfaz al objeto RepudiationAttack, que obtendrá el idioma del usuario del objeto WebSession y los textos del objeto language_parser.

El objeto RepudiationAttack eliminará el registro indicado y comprobará el nombre del usuario que lo ha eliminado. Si el alumno ha sido capaz de cambiar el nombre del usuario y que sea distinto de “Guest”, se marcará la lección como superada.

Una vez obtenido el contenido de la interfaz, el objeto screen devolverá la interfaz de respuesta al servlet para que este lo envíe a la interfaz de usuario.

4 Implementación e Implantación del Software

En esta sección se expondrán aquellos aspectos de la implementación que hayan supuesto una mayor dificultad, explicando cómo se han resuelto todos los inconvenientes encontrados.

Además se detallará un plan de pruebas que certifique que se cumplen los requisitos indicados en el apartado de análisis.

4.1 Proceso de Codificación

La implementación del código fuente de cada fichero del proyecto se ha realizado según el procedimiento indicado a continuación:

1. Creación de un nuevo paquete. Solo si es necesario y el fichero a implementar no tiene sentido en ninguno de los paquete ya existentes.
2. Creación de fichero de código Java:
    a. Indicar a qué paquete pertenece
    b. Incluir los paquetes, ficheros y clases que va a requerir el nuevo fichero para implementar su funcionalidad. Esta fase podrá completarse según se van implementando los métodos si resulta necesario incluir alguna funcionalidad de otra clase ya existente.
    c. Declarar la clase principal del fichero, indicando su visibilidad y cuyo nombre deberá coincidir con el del propio fichero.
    d. Implementar las propiedades de la clase indicadas en el diseño.
    e. Implementar el método constructor de la clase
    f. Implementar el resto de métodos necesarios para la construcción de la clase
3. Crear ficheros de contenidos que serán utilizados por la clase implementada.

A continuación se expondrán ciertos detalles de codificación que merecen ser mencionados y las pruebas para cada una de las partes del proyecto.

4.1.1 Codificación de Sistema Multilenguaje

El Sistema Multilenguaje estará formado por una clase que recuperará los textos en el idioma indicado por el usuario y los devolverá a la clase de la lección que los ha solicitado.

Se ha creado para esta clase un paquete nuevo llamado org.languages_parser, que deberá ser importado en todos los ficheros de lección para que puedan ser traducidos.

Como se ha comentado en secciones anteriores, los textos de las lecciones serán almacenados en ficheros XML, donde el sistema multilenguaje deberá localizarlos usando la librería SAX.

La librería SAX permite recorrer un documento XML secuencialmente de inicio a fin. Se ha tenido que implementar un algoritmo que detenga la lectura del documento en el nodo que se le indique, para de esta forma poder recuperar el texto que contiene.

Este algoritmo se ha implementado realizando una extensión de la clase *DefaultHandler* (que pertenece la librería SAX).

Las condiciones de parada son:

1. Encontrar un nodo con nombre “pagina” y con atributo “nombre” igual al nombre de la página buscada.
2. Si la encuentra, recorre todos los nodos con nombre “parrafo”, almacenando en una tabla Hash el atributo “posicion” y el atributo “texto”
3. Cuando SAX vuelva a leer un nodo con nombre “pagina” se detiene la lectura del documento XML.

Cada lección de WebGoat obtendrá los textos de los idiomas en el método *createContent,* heredado de la lección LessonAdapter.

4.1.2 Codificación de Clickjacking

La nueva lección Clickjacking será una extensión de la clase “LessonAdapter” y estará ubicada dentro del paquete org.owasp.webgoat.lessons.

La interfaz de usuario de la lección se ha implementado dentro del método *createContent*, dentro del cual se han codificado los cuatro elementos con los que el usuario puede interactuar:

1. Checkbox “Desactivar *iframe*”: mediante Javascript se controla que cuando se active el campo, el *iframe* situado sobre el botón “Púlsame” se sitúe por debajo de dicho botón usando la propiedad *CSS*[10] “z-index”. Cuando el

[10] CSS: Cascade Style Sheet: Hoja de estilos en cascada. Se utiliza para definir los estilos dentro de un documento HTML

checkbox queda deshabilitado, el *iframe* vuelve a situarse por encima del botón haciendo uso de la misma propiedad "z-index".

2. *Ifame*: Capa cuyo contenido es la página que el usuario verá si consigue superar la lección. Esta página llama a un fichero Javascript que es el que el alumno deberá modificar para evitar que la página se cargue en el *iframe*.
3. Botón "Púlsame": Botón que el alumno intentará pulsar. Se controla (mediante Javascript) cuándo el botón recibe un clic del ratón, entonces se muestra un mensaje por pantalla informando del evento.
4. Botón "Validar Código":Este botón realiza una recarga de pantalla, de forma que si el usuario ha realizado cambios en el código, estos se hagan efectivos.

Para detectar que el alumno ha superado la lección, se incluye un botón "confirmar" dentro de la página que va cargada en el *iframe*. De esta forma, este botón sólo será visible cuando el usuario logre que la página no se cargue en el *iframe*, si no en la página principal.

La página que contiene el *iframe,* incluye un campo oculto que indica que la lección ha sido superada. Este campo es enviado al pulsar el botón "confirmar". Al pulsar este botón se recarga la lección y el método *createContent* comprueba si ese campo existe. En caso de que exista llama al método *makeSuccess* que da la lección por superada en el sistema.

4.1.3 Codificación de Escaneo Anónimo de Puertos por CSPP

Escaneo Anónimo de Puertos por CSPP se implantará como una extensión de la clase "LessonAdapter", y estará ubicado dentro del paquete org.owasp.webgoat.lessons.

Esta lección presenta un formulario que simula una pantalla de acceso a un gestor de base de datos. El botón "Access" no envía el formulario, si no que llama a una función Javascript que comprueba las inserciones realizadas en los campos "User" y "Password".

Para comprobar que se ha superado la lección, se controla por Javascript que se haya intentado realizar una inyección por CSPP y el script que haya introducido el usuario lo haya detectado.

Este control se ha realizado utilizando otro fichero Javascript que contiene una función llamada *leccion_superada().* Esta función toma como entrada los datos que introduce el alumno en el formulario y los datos de salida que ofrece la función Javascript que debe modificar el propio alumno para evitar la inyección CSPP.

El botón "Access" del formulario de simulación tiene asociado al evento onClick la llamada a la función *leccion_superada*, de forma que al pulsarlo, si la función indica

que se ha superado la lección, se muestra un mensaje por pantalla al usuario, se establece la variable oculta *superado* con valor 1 y se recarga la lección.

Al volverse a cargar la lección, el método *createContent* detecta que la variable *superado* ha tomado valor 1, con lo que indicará al sistema que la lección ha sido superada llamando al método *makeSuccess*

4.1.4 Codificación de Repudiation Attack

La lección "Repudiation Attack" se implantará como una extensión de la clase "LessonAdapter", y se ubicará dentro del paquete org.owasp.webgoat.lessons.

La interfaz de la lección consiste en una tabla con siete registros. La columna de la izquierda que contiene un aspa roja por cada registro sirve para que al pulsarla se simule una eliminación del registro.

La simulación del borrado de registros se hace almacenando en una variable de texto de Javascript, los identificadores de las filas que van siendo borradas.

El sistema detecta que la lección ha sido superada en el método *createContent*, comprobando si el nombre de usuario que le llega es distinto del nombre de usuario que tiene la sesión abierta en el sistema. Si esta condición se cumple, se llama al método *makeSuccess*, para indicar al sistema que la lección ha sido superada.

4.2 Plan de Pruebas

Al finalizar la implementación del sistema multilenguaje y de cada lección, se realizarán pruebas sobre cada una de sus funcionalidades.

El conjunto de estas pruebas formarán el plan de pruebas del proyecto. Para superar el plan de pruebas deberán superarse todas las pruebas realizadas.

A pesar de que este plan aparece en esta sección, debe recordarse que su especificación se realiza durante la fase de análisis del proyecto.

4.2.1 Plan de pruebas para *Sistema Multilenguaje*

Las pruebas funcionales realizadas para el sistema multilenguaje se detallarán en la Tabla 36:

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Modificar Idioma	PRF01	En la pantalla de bienvenida de WebGoat, en idioma inglés, se selecciona del desplegable de idiomas la opción "Español".	Se recargará la página en español	Sí

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Cargar Lección	PRF02	Se selecciona la lección "Clickjacking" del menú de lecciones de WebGoat	Se carga la lección en el idioma seleccionado por el usuario en la prueba PRF01	Sí

Tabla 36: Pruebas Generales para Sistema Multilenguaje

4.2.2 Plan de pruebas para lección *Clickjacking*

Las pruebas funcionales realizadas para la lección de Clickjacking se especificarán en la Tabla 37

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Cargar Lección	PRF03	Se selecciona la lección de Clickjacking del menú principal	Se carga el contenido de la lección con los elementos indicados en el método createContent, los créditos indicados en el método getCredits y el nombre de la lección indicado en el método getTitle	Sí
Simulación de Clickjacking	PRF04	Se realiza un clic sobre el botón "Púlsame"	El botón no recibe el clic del ratón, lo recibe el *iframe* transparente	Sí
Activar /desactivar Clickjacking	PRF05	Marcar el campo de tipo Check "Desactivar iframe", pulsar el botón "Púlsame". Volver a desactivar el campo Check e intentar pulsar el botón nuevamente	Al seleccionar el check se mostrará un mensaje por pantalla informando que Clickjacking ha sido desactivado, el botón recibe el clic del ratón y muestra un mensaje por pantalla. Al desactivar el campo check se muestra un mensaje por pantalla informando que Clickjacking ha sido activado, el botón ya no recibe el clic del ratón	Sí
Pulsar botón "Púlsame"	PRF06	Hacer clic sobre el botón "Púlsame"	Mostrar mensaje de alerta "Clic recibido por el botón"	Sí
Validar Código Incorrecto	PRF07	Se pulsa el botón "Validar Código"	Se vuelve a cargar la lección, sin ningún cambio	Sí

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Validar Código Correcto	PRF08	Se pulsa el botón "Validar Código"	Se vuelve a cargar la lección, pero lo que se muestra es el contenido del *iframe*	Sí
Confirmar Lección Superada	PRF09	Se pulsa el botón "Confirmar"	Se vuelve a cargar la lección, indicándose en el menú un icono verde, y en el contenido de la lección un texto en rojo dando la enhorabuena porque la lección ha sido superada	Sí

Tabla 37: Pruebas Funcionales para Clickjacking

4.2.3 Plan de pruebas para lección Escaneo de Puertos por CSPP

Las pruebas funcionales realizadas para la lección de "Escaneo Anónimo de puertos por CSPP", se especificarán en la Tabla 38:

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Cargar Lección	PRF10	Se selecciona la lección de Escaneo Anónimo de puertos por CSPP del menú principal	Se carga el contenido de la lección con los elementos indicados en el método createContent, los créditos indicados en el método getCredits y el nombre de la lección indicado en el método getTitle	Sí
Acceder a la base de datos sin credenciales	PRF11	Se pulsa el botón "Access" con los campos User y Password Vacíos	Se deberá mostrar un mensaje de alerta de acceso denegado	Sí
Acceder a la base de datos con credenciales inventadas	PRF12	Se pulsa el botón "Access" completando los campos User y Password con datos inventados	Se deberá mostrar un mensaje de alerta de acceso denegado	Sí

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Acceder a la base de datos con inyección CSPP	PRF13	Se pulsa el botón "Access" completando los campos User y Password con los siguientes campos: User: **;data source= localhost, 85** Password: **;integrated security=true**	Devuelve un mensaje de alerta indicando que el puerto 85 está abierto.	Sí
Validar Código	PRF14	Se pulsa el botón "Validar Código"	Se vuelve a cargar la lección	Sí
Acceder a la base de datos con inyección CSPP y código modificado correctamente	PRF15	Se pulsa el botón "Access" completando los campos User y Password con los siguientes campos: User: **;data source= localhost, 85** Password: **;integrated security=true**	Se muestra un mensaje de alerta indicando que se ha superado la lección y se vuelve a cargar la lección, indicándose en el menú un icono verde, y en el contenido de la lección un texto en rojo dando la enhorabuena porque la lección ha sido superada	Sí

Tabla 38: Pruebas Funcionales para Escaneo Anónimo de Puertos

4.2.4 Plan de pruebas para lección *Repudiation Attack*

Las pruebas funcionales realizadas para la lección de "Repudiation Attack", se especificarán en la Tabla 39:

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Cargar Lección	PRF16	Se selecciona la lección de Repudiation Attack del menú principal	Se carga el contenido de la lección con los elementos indicados en el método createContent, los créditos indicados en el método getCredits y el nombre de la lección indicado en el método getTitle. La tabla de registros se muestra completa al cargar la lección.	Sí

Nombre	Código	Datos de Entrada	Resultado Esperado	Superada
Borrar Registro	PRF17	Se pulsa sobre la "X" roja del registro correspondiente	Se deberá volver a cargar la lección sin el registro y un mensaje con el nombre del usuario que ha borrado el registro	Sí
Superar Lección	PRF18	Se pulsa sobre la "X" roja de un registro y se modifica mediante un analizador de redes, el nombre del usuario	Se deberá volver a cargar la lección sin el registro y un mensaje con el nombre de usuario que se ha indicado en el analizador de redes.	Sí

Tabla 39: Pruebas Funcionales para Repudiation Attack

4.2.5 Resultado de Plan de Pruebas

Al haberse superado todas las pruebas incluidas en el plan se da por superado el plan de pruebas del proyecto.

5 Conclusiones y Líneas Futuras

Este capítulo contiene las principales conclusiones alcanzadas tras la realización del proyecto, así como las líneas futuras de trabajo que se identifican tras la finalización del mismo.

5.1 Conclusiones Sobre el Proyecto

En esta sección se expondrán algunas de las impresiones sobre el presente Proyecto, estructuradas en distintos apartados.

5.1.1 Resultado Obtenido

El producto obtenido tras la elaboración del proyecto se describiría como una versión mejorada de WebGoat preparada para que una parte significativa de su contenido pueda ser leída en inglés y castellano.

Además permitirá la traducción en otros idiomas de forma sencilla, simplemente se tendrá que traducir el fichero XML que contiene los textos de los idiomas en el idioma deseado.

Las nuevas lecciones que se han implementado amplían y mejoran la aplicación, actualizándola para evitar que se quede obsoleta.

5.1.2 Dificultad del Proyecto

A lo largo de la realización del Proyecto se han identificado cuatro aspectos que han supuesto una dificultad singular para la realización del proyecto. A continuación se presentan dichos aspectos.

En primer lugar, debe destacarse la falta de experiencia con las herramientas de desarrollo. Este hándicap ha resultado determinante en las fases de diseño y codificación del proyecto, ya que se ha requerido mucho tiempo para consultar sintaxis, funcionalidades y estructuras elementales de Java y sobre el funcionamiento de la plataforma J2EE.

En segundo lugar, la existencia de código heredado ha requerido de un tiempo extra para comprender su funcionamiento actual y diseñar la mejor estrategia para su ampliación.

En tercer lugar, debido a los compromisos laborales del autor las horas empleadas para el desarrollo del proyecto de lunes a viernes han sido las últimas del día, factor que ha provocado un descenso notable en la productividad.

Finalmente, gran parte del tiempo empleado en el proyecto ha sido invertido en investigar. Las estimaciones de tiempo para la investigación son difíciles de calcular, ya

que no se pueden definir con exactitud etapas cuantificables para completar este proceso.

5.1.3 Desarrollo del Proyecto

En las fases de análisis y diseño ha resultado crítico el tiempo empleado en el estudio del código existente en WebGoat, ya que sin un preciso conocimiento del mismo resultaba imposible integrar las mejoras planteadas y que suponen el objeto del presente proyecto.

Para la implementación de las soluciones propuestas se ha necesitado emplear gran cantidad de tiempo en consultar y estudiar los funcionamientos del entorno de desarrollo para que la programación realizada genere un producto eficaz y, en lo posible, eficiente.

En la fase de pruebas se hace notorio el trabajo realizado en las fases anteriores, ya que se ve reflejado que cuanto mejor se realizan las fases previas, se dan menos errores en el resultado final, además de resultar mucho más fácil las modificaciones y el futuro mantenimiento.

Inicialmente se intentaron implementar las nuevas funcionalidades sobre la versión 5.3 de WebGoat, pero después de una semana de investigación para intentar crear un entorno de desarrollo, se desestimó porque no se obtuvieron resultados positivos.

La versión 5.2 de WebGoat dispone de una versión para programadores con un entorno de desarrollo preparado para ser utilizado. Contiene todas las funcionalidades de la versión 5.2 con una versión de eclipse pre-configurada para comenzar a trabajar.

5.1.4 Gestión Temporal y Económica del Proyecto

Ha resultado evidente en la elaboración del proyecto, que la estimación de tiempos inicial tiene un gran impacto sobre el resultado económico final del proyecto.

Como se ha podido comprobar, la valoración de tiempos afecta directamente en la planificación a la que se atienen todos los recursos que intervienen en el proyecto, si los recursos precisan de más tiempo, requerirán un esfuerzo económico mayor.

El retraso de los tiempos estimados, causados por el personal encargado de la elaboración del proyecto, hace que el valor del mismo se vea altamente afectado y sea muy inferior al calculado al inicio del proyecto.

También es destacado el porcentaje económico del proyecto que se destina a la Seguridad Social y a Hacienda (Agencia Tributaria), factores que deben tomarse muy en cuenta para obtener una rentabilidad aceptable.

5.2 Líneas Futuras

El presente proyecto puede ampliarse en varios sentidos, que se detallarán a continuación:

1. Ampliar los idiomas del contenido de la aplicación. El diseño del sistema multilenguaje permite que la traducción a cualquier idioma resulte un proceso sencillo, puesto que para cada idioma existe un fichero que contiene todos los textos de WebGoat, por tanto bastaría con traducir uno de estos ficheros de idioma al idioma deseado.
2. Incluir nuevas lecciones. Será necesario mantener el sistema actualizado con los últimos ataques aparecidos contra aplicaciones Web.
3. Adaptarlo a la tecnología AJAX[11]. De forma que resulte una aplicación más cómoda para el cliente
4. Utilizar librerías de Javascript como jQuery que faciliten y mejoren la codificación del lado del cliente.
5. Colaborar con el equipo OWASP para la incorporación de estas mejoras (multilenguaje y nuevas lecciones) en la versión 5.3 de WebGoat, contribuyendo de esta forma a la estabilización y consolidación de la nueva versión.

[11] AJAX: Tecnología que permite que el cliente interactúe con el servidor sin necesidad de recargar la página entera en el navegador.

6 Bibliografía y Referencias

Agencia Tributaria. (s.f.). *AEAT*. Obtenido de http://www.aeat.es

Alfa-Risk SL Correduría de Seguros. (s.f.). *Seguro Responsabilidad Civil para informáticos autónomos*. Recuperado el Diciembre de 2010, de http://www.segurosinformatica.es/seguro-responsabilidad-civil-informaticos.asp

ConnectionStrings.com. (s.f.). *ConnectionStrings.com*. Recuperado el Diciembre de 2010, de http://www.connectionstrings.com/

ESA Comité de Estandarización y Control de Software. (Julio de 2003). Guía para la aplicación de Estándares de Ingeniería de Software ESA para proyectos de software pequeños. París, Francia.

Facebook. (Mayo de 2010). *All about Clickjacking*. Recuperado el Febrero de 2011, de http://www.facebook.com/topic.php?uid=296343844022&topic=15380

Internet Assigned Numbers Authority (IANA). (s.f.). *Port Numbers Assignments*. Recuperado el Febrero de 2011, de http://www.iana.org/assignments/port-numbers

Mapfre. (s.f.). *Plan Integral Autónomos de Mapfre*. Recuperado el 12 de 2010, de http://www.mapfre.com/seguros/es/profesionales/soluciones/plan-integral-autonomos.shtml

Ministerio de Trabajo e Inmigración, Gobierno de España. (s.f.). *Seguridad Social*. Obtenido de http://www.seg-social.es

Oracle Corporation. (s.f.). *Java 2 Platform, Enterprise Edition (J2EE) Overview*. Obtenido de http://java.sun.com/j2ee/overview.html

OWASP. (Diciembre de 2001). *Open Web Application Security Project*. Obtenido de http://www.owasp.org

OWASP. (s.f.). *WebGoat Project.* Obtenido de http://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

RAE. (s.f.). *Real Academia Española*. Obtenido de www.rae.es

Saxproject. (s.f.). *Saxproject*. Obtenido de http://www.saxproject.org

The Apache Software Foundation. (s.f.). *Apache Tomcat*. Obtenido de http://tomcat.apache.org/

The hsql Development Group. (s.f.). *HSQLDB*. Obtenido de http://hsqldb.org/

7 ANEXO I: Gestión del Proyecto

Para la realización de este proyecto se precisará el seguimiento de una planificación basada en estimaciones de factores temporales y económicos que se detallarán en esta sección.

7.1 Planificación del Trabajo

Para abarcar el presente proyecto de forma viable, será necesario alcanzar un grado de abstracción lo suficientemente alto como para poder esclarecer y dividir el objetivo general en grupos de objetivos que a su vez se podrán dividir en más subgrupos, tantas veces como sea necesario hasta llegar a objetivos fácilmente abarcables. En definitiva, se hará uso de la técnica “divide y vencerás” para definir todas las tareas a realizar.

El objetivo principal del proyecto es desarrollar una nueva versión multilenguaje de la aplicación WebGoat ampliando las lecciones existentes.

De este objetivo se pueden deducir dos sub-objetivos:

1. Implementar sistema multilenguaje para WebGoat: Para abordar este objetivo se necesitarán realizar las siguientes tareas:
    a. Estudio de la arquitectura de WebGoat: Estructura de ficheros, relación entre clases, entradas y salidas de datos etc.
    b. Análisis de las posibles soluciones.
    c. Elección de la solución adecuada
    d. Diseño de la solución escogida.
    e. Implementación de la solución.
    f. Carga de datos.
    g. Pruebas
2. Implementar nuevas lecciones para WebGoat. Se deberán realizar las siguientes tareas:
    a. Estudio de las lecciones existentes en WebGoat
    b. Búsqueda de ataques no contemplados en las lecciones de WebGoat
    c. Estudio de viabilidad de los ataques buscados para ser implementados como lecciones en WebGoat.
    d. Selección de las lecciones a implementar
    e. Por cada lección nueva a implementar será necesario realizar las siguientes tareas:
        i. Análisis del ataque objeto de la lección.
        ii. Establecer los requisitos para la lección
        iii. Análisis de los requisitos establecidos
        iv. Diseño de una solución que satisfaga los requisitos.

v. Implementación del diseño
vi. Pruebas
vii. Añadir el contenido teórico de la lección

La Ilustración 31 ofrece una representación gráfica de la estructura jerárquica de la abstracción de las tareas identificadas:

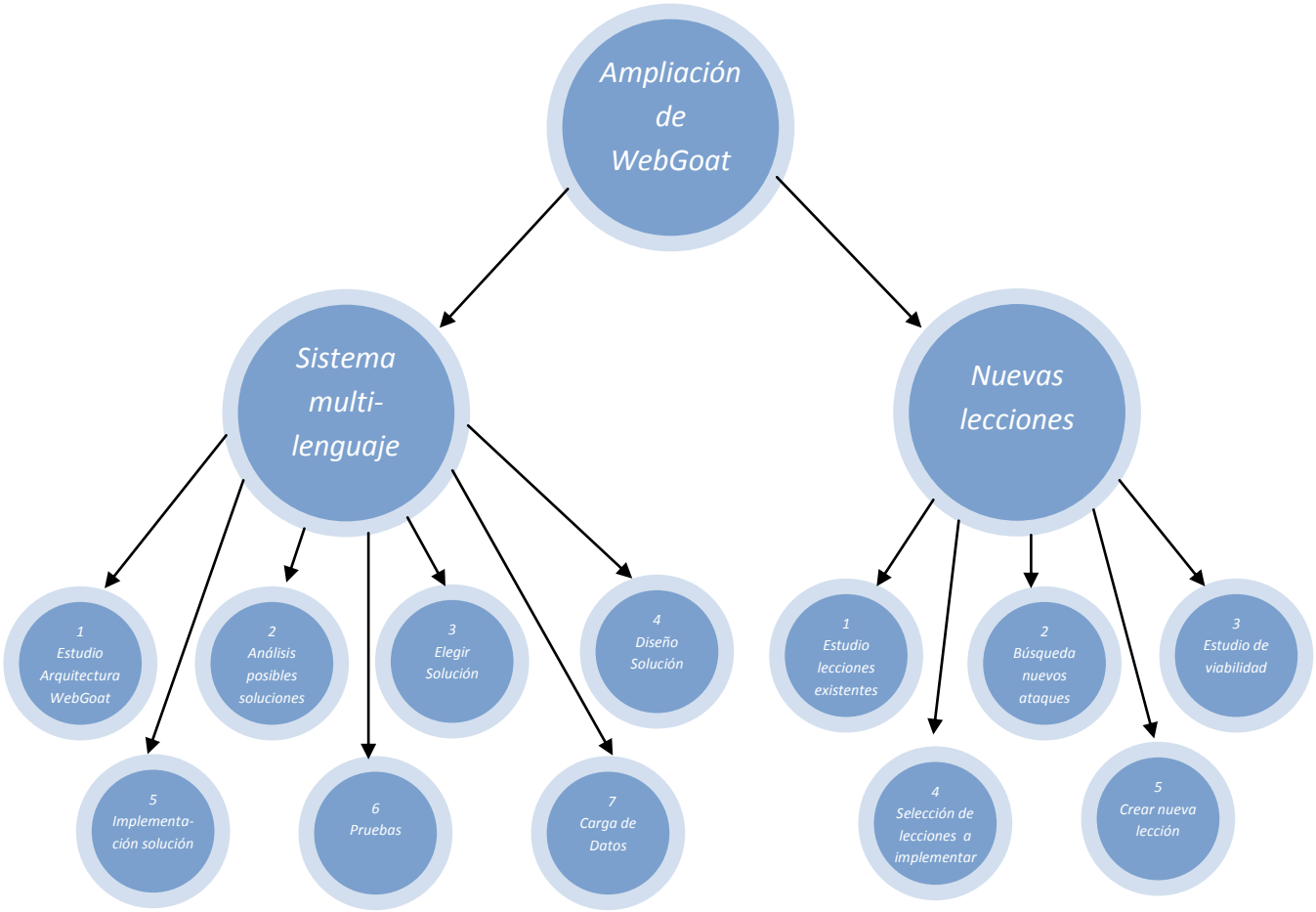

Ilustración 31 Esquema de tareas

Cada nivel de esta estructura jerárquica representará un grado de abstracción, siendo el nodo padre, el grado más abstracto y los nodos hijos los más concretos.

Para una mejor comprensión, las tareas para crear una lección se muestran gráficamente en la Ilustración 32:

Ilustración 32 Esquema de tareas para crear lección

La numeración de los nodos concretos atiende al orden en que deberán realizarse las tareas.

Las tareas que se han tratado hasta ahora son las referidas únicamente al desarrollo del proyecto, pero también deben tenerse en cuenta las tareas de instalación y adecuación del entorno de trabajo y las de documentación, que se detallarán en los siguientes apartados.

En la elaboración del proyecto se dispondrá de un solo recurso para realizar todas las tareas.

7.1.1 Planificación Inicial

Al realizar un primer estudio sobre el proyecto a desarrollar se propone seguir un ciclo de vida iterativo e incremental.

En la introducción de este apartado se han obtenido las tareas a realizar para lograr los objetivos propuestos, por lo que se deberá adecuar dichas tareas al modelo de ciclo de vida propuesto.

Cada tarea se realizará en un determinado número de ciclos, de forma que en cada ciclo se aporten elementos que amplíen lo realizado en el ciclo anterior hasta alcanzar el objetivo final de la tarea.

El número de ciclos dependerá, por tanto de la tarea, pero se estima que, atendiendo al tipo de tarea se precisarán aproximadamente un número de ciclos determinado, tal y como se indica a continuación:

1. Tareas de concepción: bastará con **una sola iteración** para definir los requisitos y objetivos.
2. Tareas de elaboración: abarcarán las tareas de análisis y diseño, y para este tipo de proyecto bastarían **máximo dos iteraciones** para cada tarea, en este caso dependerá de la tarea concreta.
3. Tareas de construcción: para este tipo de tareas es más complicado estimar un número fijo de iteraciones, pero no deberían requerir menos de dos, ya que las tareas de programación convienen segmentarlas al máximo para facilitar posibles modificaciones y mantenimientos futuros.

Las razones por las que se elige este ciclo de vida son:

1. Reducir el coste del riesgo a los costes de un solo incremento. Si se tiene que repetir la iteración, sólo se perderá el esfuerzo mal empleado de dicha iteración, no el valor del producto entero
2. Reducir el riesgo de retrasar la entrega del proyecto por errores en las fases tempranas. Este modelo facilita la detección de posibles errores en fases iniciales, de forma que evita que al finalizar el proyecto se detecten fallos que requieran una reestructuración inviable por haber superado el tiempo de entrega.
3. Obtener resultados claros a corto plazo. Permitirá acelerar el ritmo del esfuerzo de desarrollo al comprobar que se está trabajando de un modo eficiente.
4. Los requisitos se podrán refinar en cada iteración, de esta forma no se tendrá la necesidad de definir los requisitos de forma precisa en las fases iniciales.

En las ilustraciones que se mostrarán a continuación se podrán apreciar los gráficos Gantt de las tareas a realizar. Para una mejor comprensión se mostrará en varios gráficos distintos, para poder tener una visión general y concreta de las tareas a realizar.

Para ayudar en la interpretación de los diagramas de Gantt, se explicarán los elementos de los que se componen:

1. Nombre de tarea: Columna que contendrá un texto descriptivo que nos servirá para identificar cada una de las tareas que forman parte del diagrama.

2. Duración: Indicará la duración de cada una de las tareas. La duración de las tareas Resumen se medirán en días y las tareas concretas se medirán en horas.
3. Regla de tiempo: contendrá una referencia temporal para representar en el diagrama la duración de las tareas. Dependiendo del tamaño del diagrama y para una mejor comprensión, la regla de tiempo aparecerá en distintas escalas (cada 7 días, cada 4 días, cada mes, cada trimestre …)
4. Tareas resumen: estarán representadas en el gráfico por una línea negra  , que indicará en el gráfico la duración total de todas las tareas concretas agrupadas en la tarea resumen.
5. Tareas concretas: estarán representadas en el gráfico por una línea azul  , que representará en el gráfico la duración de la tarea.

Las estimaciones se han realizado en base a la participación de un solo recurso en la elaboración de todo el proyecto.

7.1.1.1.1 Estimación Global del Proyecto

El diagrama mostrado en la Ilustración 33 contendrá una visión global de la duración estimada inicialmente para proyecto.

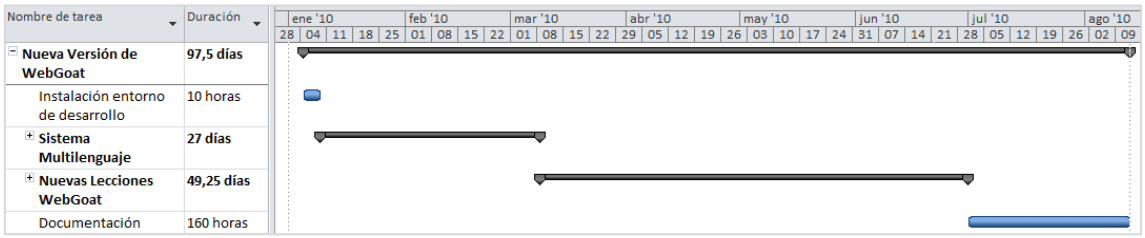

Ilustración 33 Diagrama de Gantt – Estimación global del proyecto

Según la estimación realizada del proyecto, se obtendrán los resultados indicados en la Tabla 40:

Fecha de inicio del proyecto	04/01/2010
Fecha de finalización del proyecto	12/08/2010
Duración en días del proyecto	97,50 días
Horas totales empleadas	780 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 40: Estimación global inicial de tiempos

7.1.1.1.2 Estimación para "Sistema Multilenguaje"

En la Ilustración 34 se presentará el diagrama de Gantt con las tareas agrupadas de "Sistema Multilenguaje".

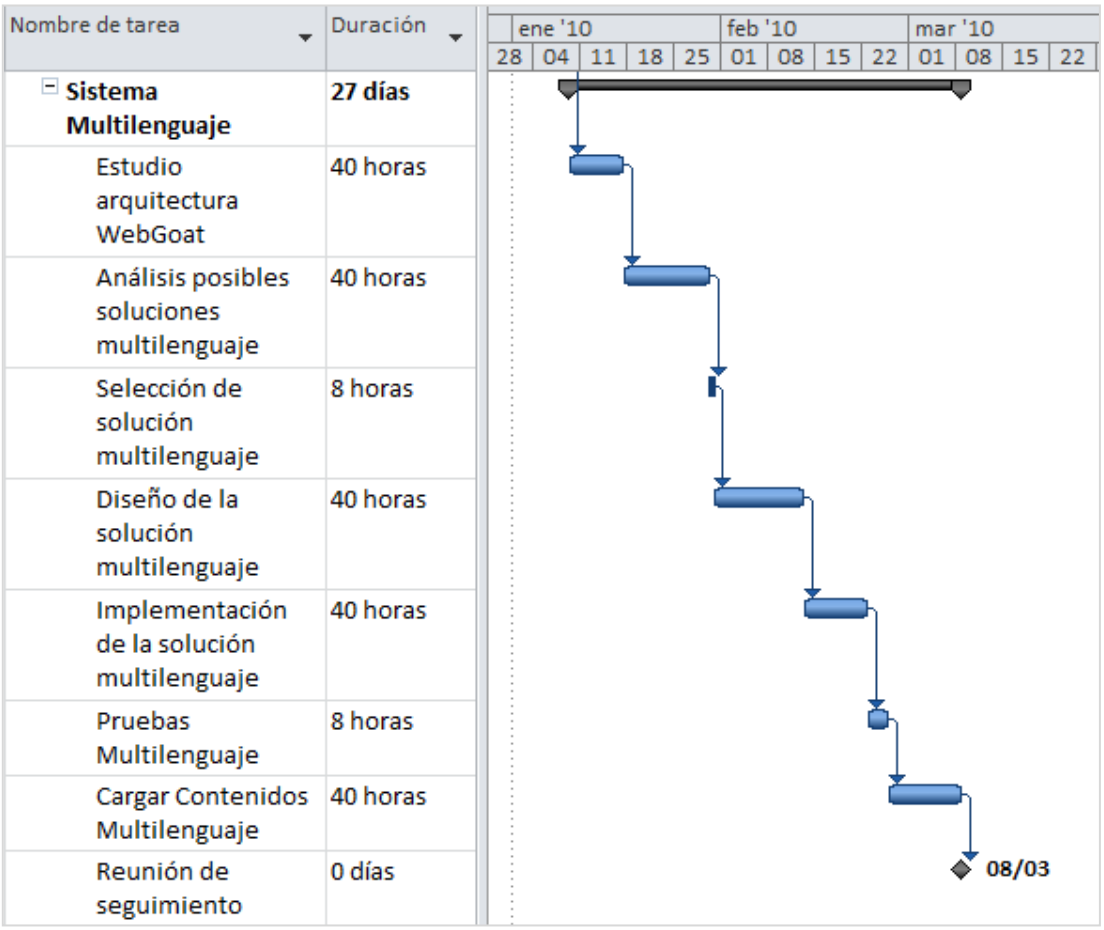

Ilustración 34 Diagrama de Gantt – Estimación de tiempos para "Sistema Multilenguaje"

Según la estimación realizada para "Sistema Multilenguaje", se obtendrán los resultados indicados en la Tabla 41:

Fecha de inicio de "Sistema Multilenguaje"	09/01/2010
Fecha de finalización de "Sistema Multilenguaje"	18/03/2010
Duración en días de "Sistema Multilenguaje"	27 días
Horas totales empleadas	216 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 41: Estimación de tiempos para "Sistema Multilenguaje"

7.1.1.1.3 Estimación para "Nuevas Lecciones WebGoat"

La estimación para las nuevas lecciones se detallará en la Ilustración 35

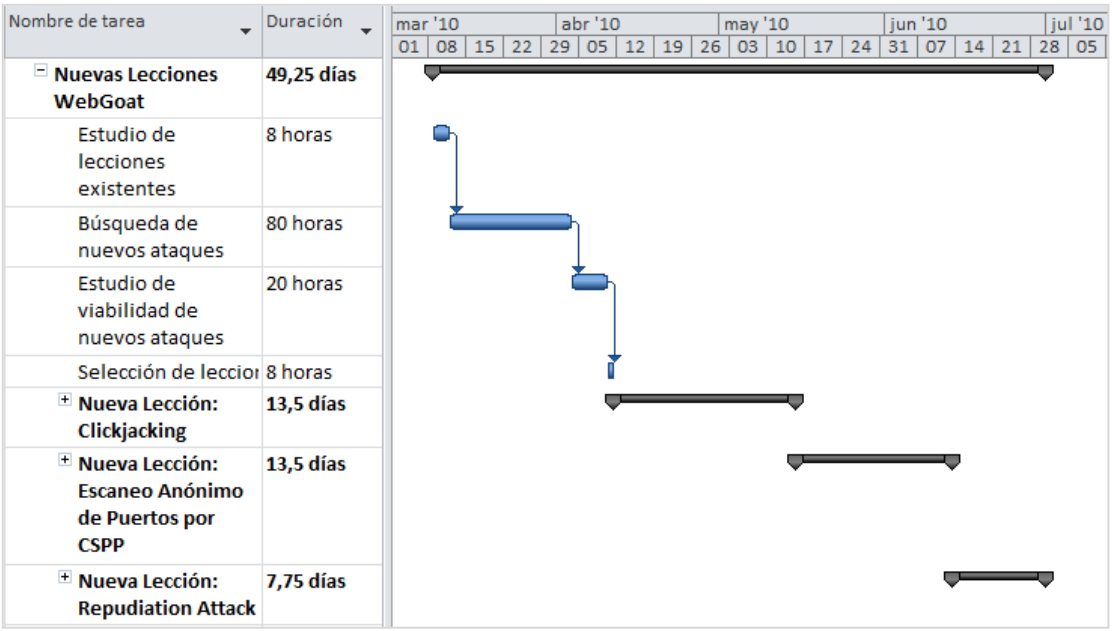

Ilustración 35 Diagrama de Gantt - Estimación de tiempos para "Nuevas Lecciones WebGoat"

Para la elaboración de nuevas lecciones será necesario realizar un estudio previo que conste de una seria de tareas que permitan estudiar y escoger las lecciones a implementar.

Cada nueva lección se indica como tarea resumen, y será explicada en los diagramas siguientes.

Según la estimación realizada para "Nuevas Lecciones", se obtendrán los resultados indicados en la Tabla 42:

Fecha de inicio de "Nuevas Lecciones"	08/03/2010
Fecha de finalización de "Nuevas Lecciones"	30/06/2010
Duración en días de "Nuevas Lecciones"	49,25 días
Horas totales empleadas	278 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 42: Estimación de tiempos para "Nuevas Lecciones WebGoat"

Las estimaciones para cada una de las nuevas lecciones se presentarán a continuación.

7.1.1.1.4 Estimación para "Clickjacking"

El diagrama de Gantt de las tareas para la lección de "Clickjacking" se detallará como sigue

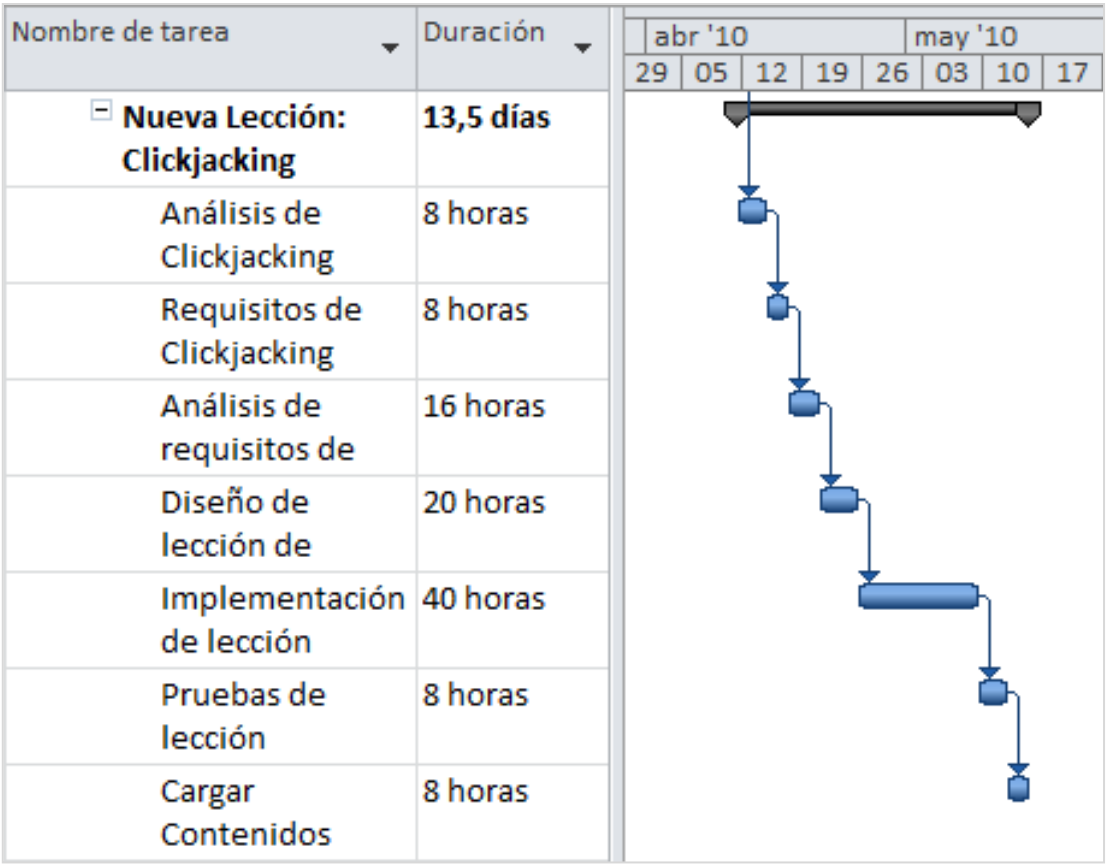

Ilustración 36 Diagrama de Gantt - Estimación de tiempos para "Clickjacking"

Según la estimación realizada para la lección "Clickjacking", se obtendrán los resultados indicados en la Tabla 43:

Fecha de inicio de "Clickjacking"	11/04/2010
Fecha de finalización de "Clickjacking"	15/05/2010
Duración en días de "Clickjacking"	13,50 días
Horas totales empleadas	108 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 43: Estimación de tiempos para "Clickjacking"

7.1.1.1.5 Estimación para "Escaneo Anónimo de Puertos por CSPP"

En la Ilustración 37 se mostrará el diagrama de Gantt correspondiente a la lección "Escaneo Anónimo de Puertos por CSPP"

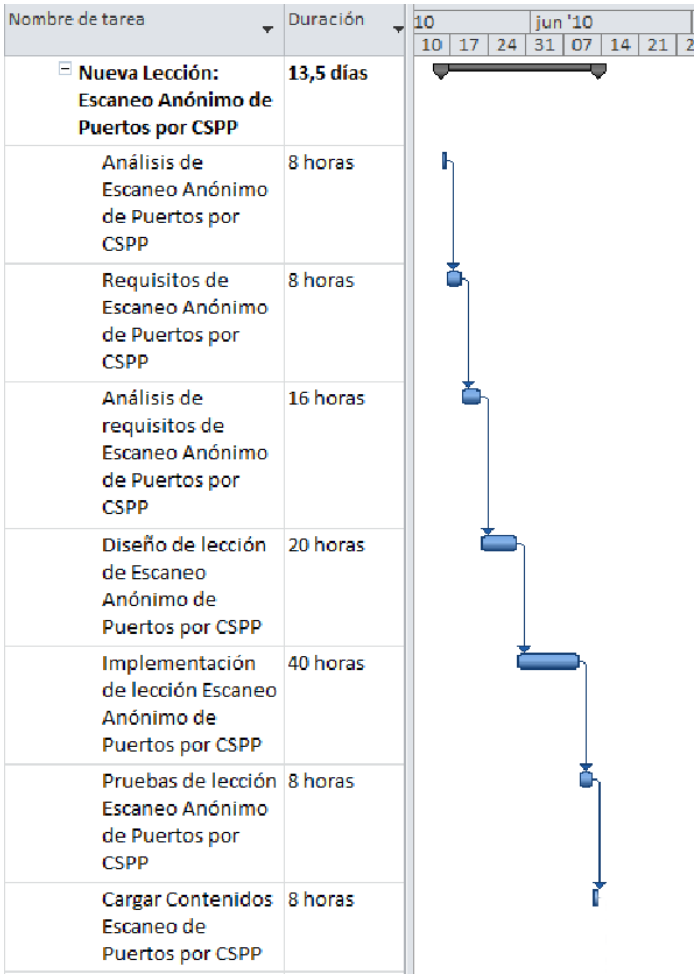

Ilustración 37 Diagrama de Gantt - Estimación de tiempos para "Escaneo Anónimo de Puertos por CSPP"

Según la estimación realizada para "Escaneo Anónimo de Puertos por CSPP", se obtendrán los resultados indicados en la Tabla 44:

Fecha de inicio de "Escaneo Anónimo de Puertos por CSPP"	15/05/2010
Fecha de finalización de "Escaneo Anónimo de Puertos por CSPP"	13/06/2010
Duración en días de "Escaneo Anónimo de Puertos por CSPP"	13,50 días
Horas totales empleadas	108 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 44: Estimación de tiempos para "Escaneo Anónimo de Puertos por CSPP"

7.1.1.1.6 Estimación para "Repudiation Attack"

El diagrama que se presentará en la Ilustración 38 reflejará la estimación de tiempos para crear la lección "Repudiation Attack"

Nombre de tarea	Duración
Nueva Lección: Repudiation Attack	**7,75 días**
Análisis de Repudiation Attack	8 horas
Requisitos de Repudiation Attack	4 horas
Análisis de requisitos de Repudiation Attack	8 horas
Diseño de lección de Repudiation Attack	10 horas
Implementación de lección Repudiation Attack	20 horas
Pruebas de lección Repudiation Attack	4 horas
Cargar Contenidos Repudiation Attack	8 horas
Reunión de seguimiento	0 días

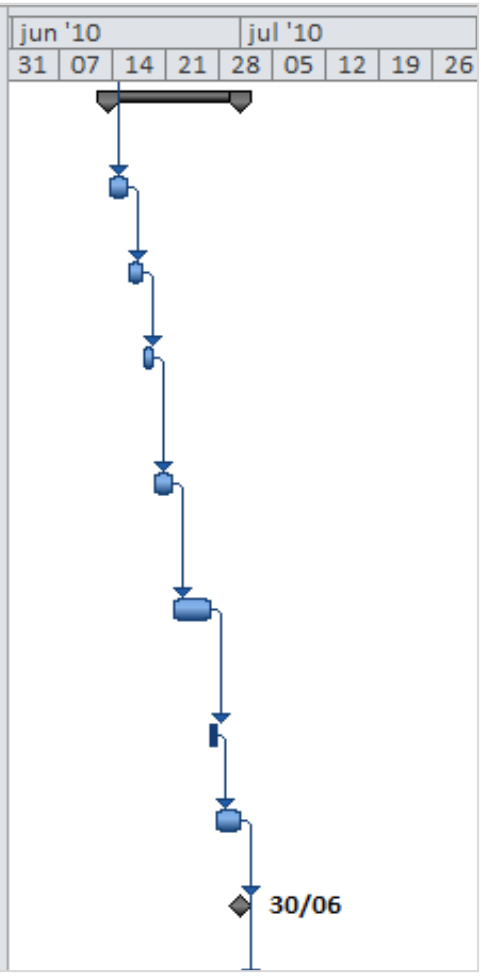

Ilustración 38 Diagrama de Gantt - Estimación de tiempos para "Repudiation Attack"

Según la estimación realizada para "Repudiation Attack", se obtendrán los resultados indicados en la Tabla 45:

Fecha de inicio de "Repudiation Attack"	13/06/2010
Fecha de finalización de "Repudiation Attack"	30/06/2010
Duración en días de "Repudiation Attack"	7,75 días
Horas totales empleadas	62 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 45: Estimación de tiempos para "Repudiation Attack"

7.1.2 Desarrollo Real del Proyecto

Finalizado el proyecto, se han obtenido los datos reales de la planificación, pudiendo obtener conclusiones sobre el éxito o fracaso de la planificación inicial.

A continuación se expondrán los diagramas de Gantt del apartado anterior, pero reajustado al tiempo real de desarrollo.

7.1.2.1.1 Duración Real del Proyecto

La duración real del proyecto quedará reflejada en la Ilustración 39

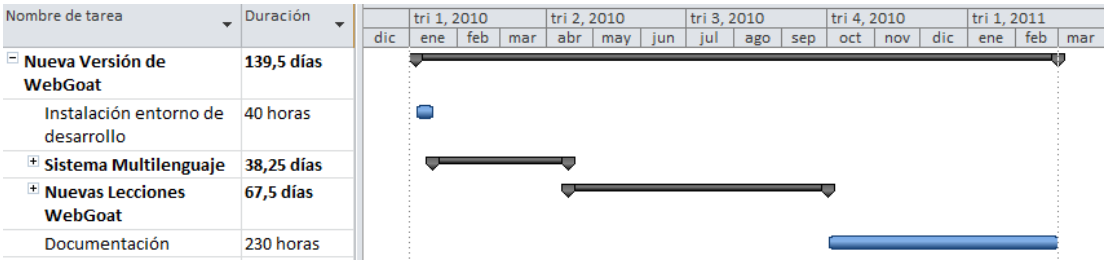

Ilustración 39 Diagrama de Gantt - Duración real del Proyecto

Del diagrama de Gantt correspondiente a la visión global del proyecto, se obtendrán los resultados indicados en la Tabla 46:

Fecha de inicio del proyecto	04/01/2010
Fecha de finalización del proyecto	28/02/2011
Duración en días del proyecto	139,5 días
Horas totales empleadas	1108 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 46: Duración real del Proyecto

Desde un punto de vista global, se aprecia que la duración real del proyecto ha superado en 238 horas la estimación inicial que se ha realizado.

En la Tabla 47 se muestra una comparativa de tiempos

	Tiempo Estimado	**Tiempo Real**
Fecha de inicio del proyecto	**04/01/2010**	**04/01/2010**
Fecha de finalización del proyecto	12/08/2010	04/12/2010
Duración en días del proyecto	97,50 días	139,5 días
Horas totales empleadas	780 horas	1108 horas

Tabla 47: Comparativa de tiempos

Las causas se han debido principalmente a dos factores:

1. Estimación de tiempos: El número de horas que en un principio se asigno a las tareas fue inferior al tiempo real necesitado. Las tareas que más han sufrido este retraso son:
    a. Instalación del entorno de desarrollo: La estimación inicial se realizó contando con que se podría instalar en Ubuntu de una forma intuitiva. La realidad fue que se intentó instalar la versión de WebGoat para desarrolladores para Ubuntu, pero al superar un periodo de tiempo aceptable se decidió instalarlo con la versión para Windows.
    b. Tareas de Diseño e Implementación: Al desconocer la plataforma y el lenguaje de programación, se empleo mucho tiempo buscando y consultando clases y métodos necesarios para el desarrollo del proyecto.
2. Rendimiento: Las estimaciones iniciales se realizaron teniendo en cuenta que el recurso tendría un rendimiento de un 100%. La realidad fue que en la mayoría de las jornadas, el rendimiento sería de un 80%, llegando en algunos casos a no superar el 50%, puesto que las horas dedicadas al proyecto eran las finales del día, habiendo dedicado el resto de la jornada a la realización de otros proyectos.

7.1.2.1.2 Duración Real de "Sistema Multilenguaje"

La Ilustración 40 se muestra la duración de las tareas de "Sistema Multilenguaje"

Nombre de tarea	Duración
Sistema Multilenguaje	**38,25 días**
Estudio arquitectura WebGoat	40 horas
Análisis posibles soluciones multilenguaje	40 horas
Selección de solución multilenguaje	8 horas
Diseño de la solución multilenguaje	70 horas
Implementación de la solución multilenguaje	100 horas
Pruebas Multilenguaje	8 horas
Cargar Contenidos Multilenguaje	40 horas
Reunión de seguimiento	0 días

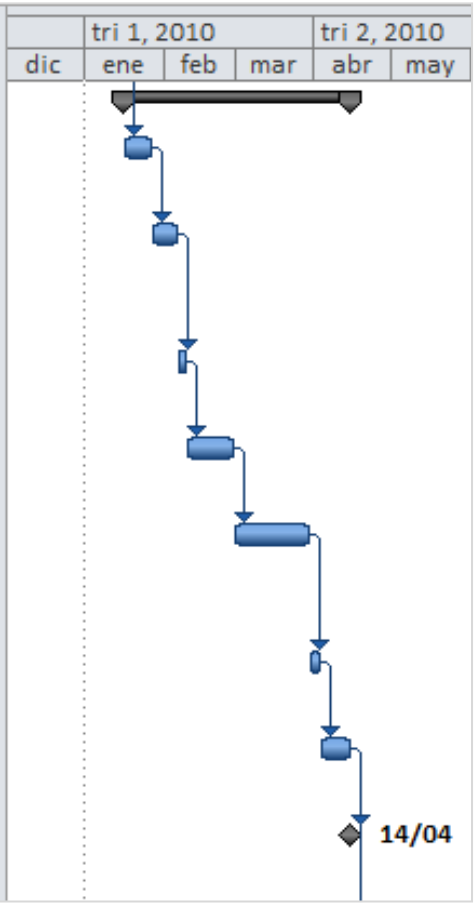

Ilustración 40 Diagrama de Gantt - Duración de tareas "Sistema Multilenguaje"

Del diagrama de Gantt correspondiente al "Sistema Multilenguaje", se han obtenido los resultados indicados en la Tabla 48:

Fecha de inicio de "Sistema Multilenguaje"	16/01/2010
Fecha de finalización de "Sistema Multilenguaje"	14/04/2010
Duración en días de "Sistema Multilenguaje"	38,25 días
Horas totales empleadas	306 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 48: Duración de tareas "Sistema Multilenguaje"

7.1.2.1.3 Duración Real de "Nuevas Lecciones WebGoat"

La duración de las tareas para las nuevas lecciones se detallará en la Ilustración 41

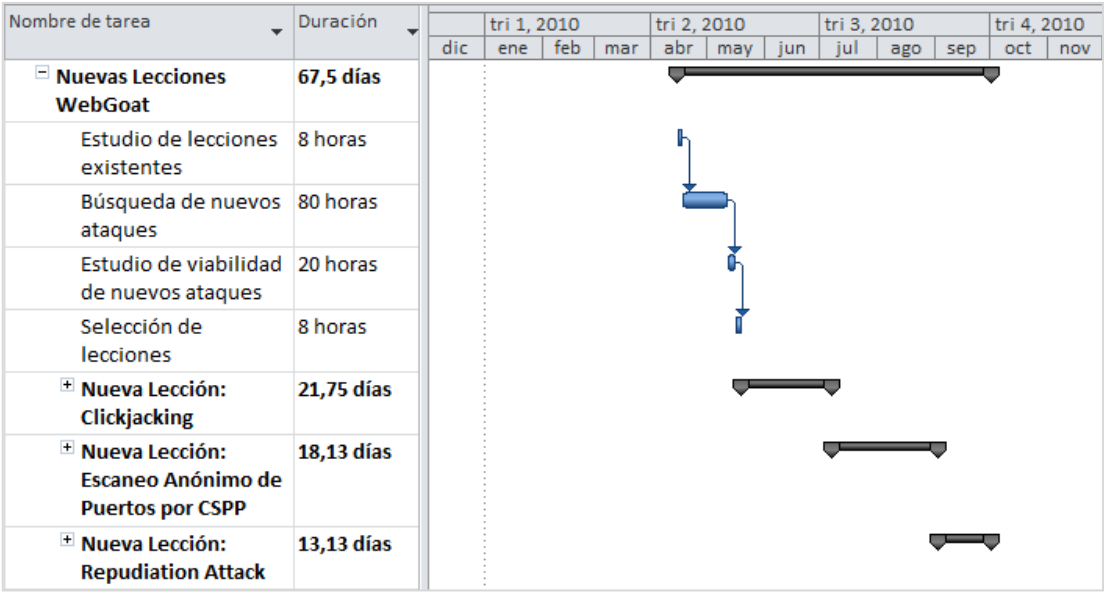

Ilustración 41 Diagrama de Gantt - Duración de tareas "Nuevas Lecciones WebGoat"

Del diagrama de Gantt correspondiente a "Nuevas Lecciones WebGoat", se obtendrán los resultados indicados en la Tabla 49:

Fecha de inicio de "Nuevas Lecciones WebGoat"	14/04/2010
Fecha de finalización de "Nuevas Lecciones WebGoat"	01/10/2010
Duración en días de "Nuevas Lecciones WebGoat"	67,5 días
Horas totales empleadas	540 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 49: Duración de tareas "Nuevas Lecciones WebGoat"

7.1.2.1.4 Duración Real de "Clickjacking"

Las tareas realizadas para crear la lección de "Clickjacking" y la duración de las mismas se expondrán en el diagrama de Gantt contenido en la Ilustración 42

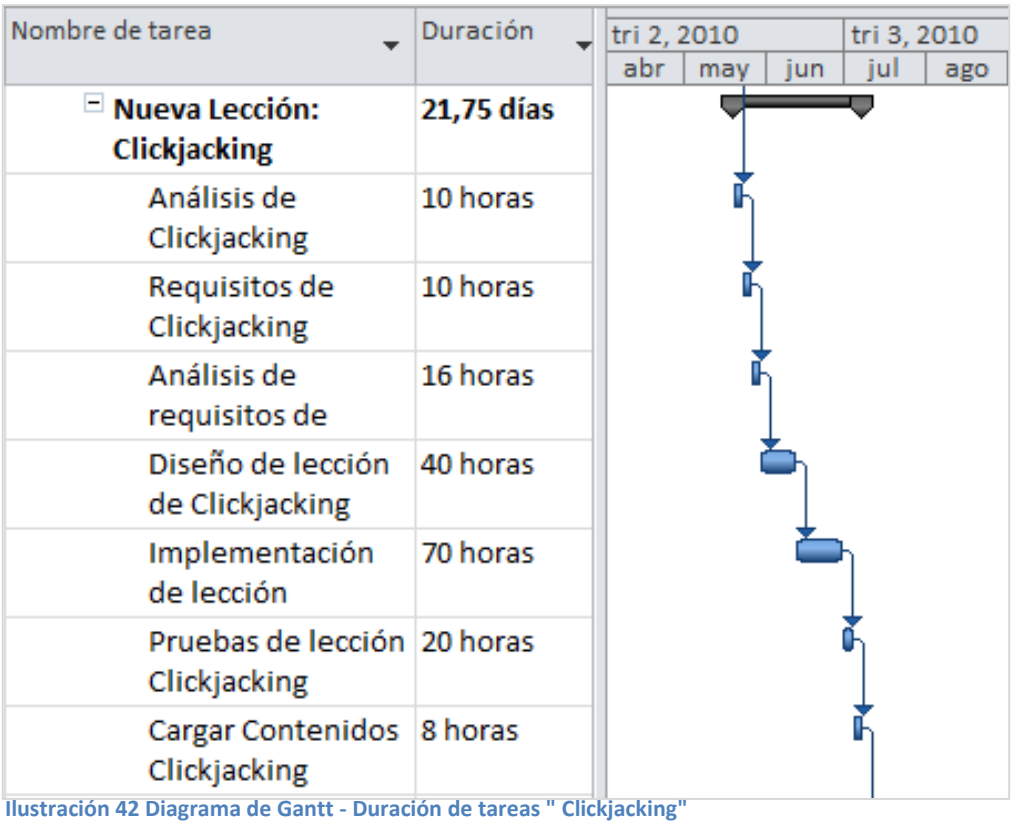

Ilustración 42 Diagrama de Gantt - Duración de tareas " Clickjacking"

Del diagrama de Gantt correspondiente a "Clickjacking", se obtendrán los resultados indicados en la Tabla 50:

Fecha de inicio de "Clickjacking"	19/05/2010
Fecha de finalización de "Clickjacking"	07/07/2010
Duración en días de "Clickjacking"	21,75 días
Horas totales empleadas	174 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 50: Duración de tareas "Clickjacking"

7.1.2.1.5 Duración Real de "Escaneo Anónimo de Puertos por CSPP"

La duración de las tareas para la creación de la lección de "Escaneo Anónimo de Puertos por CSPP" se detallará en la Ilustración 43

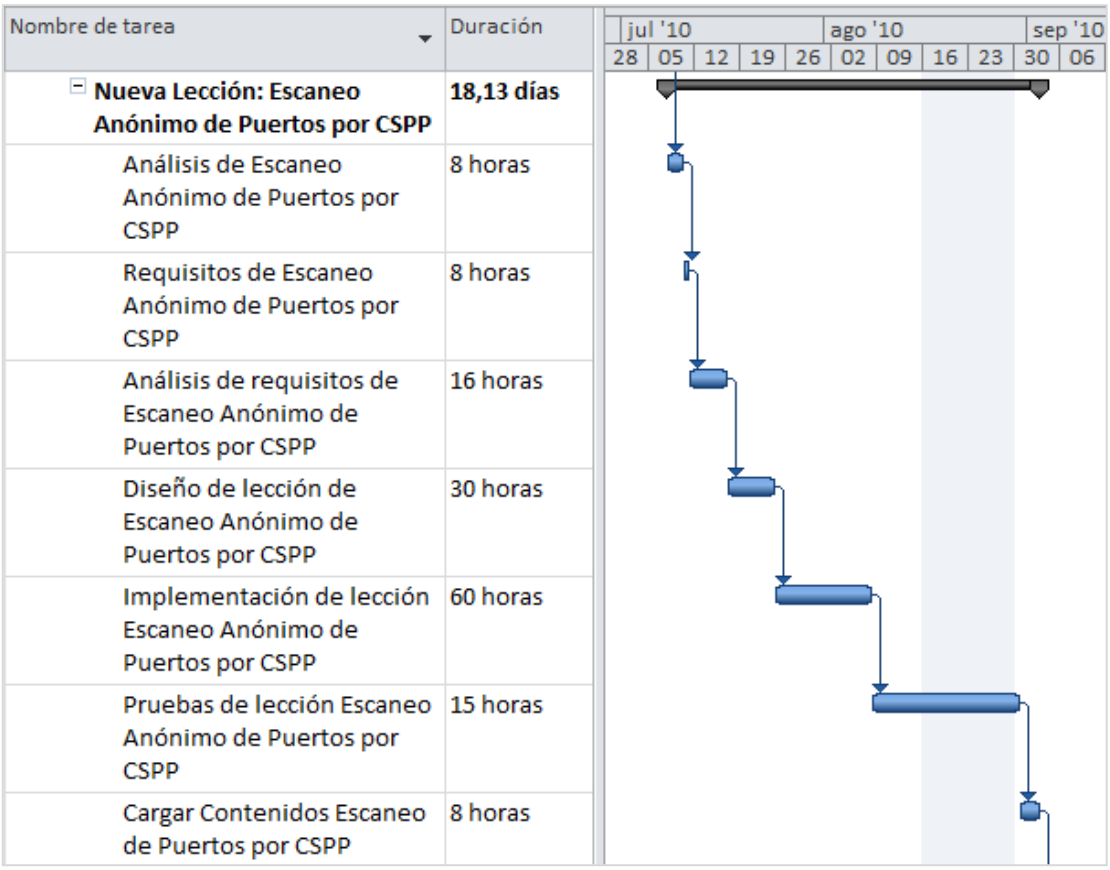

Ilustración 43 Diagrama de Gantt - Duración de tareas "Escaneo Anónimo de Puertos por CSPP"

Del diagrama de Gantt correspondiente a "Escaneo Anónimo de Puertos por CSPP", se obtendrán los resultados indicados en la Tabla 51:

Fecha de inicio de "Escaneo Anónimo de Puertos por CSPP"	07/07/2010
Fecha de finalización de "Escaneo Anónimo de Puertos por CSPP"	02/09/2010
Duración en días de "Escaneo Anónimo de Puertos por CSPP"	18,13 días
Horas totales empleadas	145 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 51: Duración de tareas "Escaneo Anónimo de Puertos por CSPP"

7.1.2.1.6 Duración Real de "Repudiation Attack"

La Ilustración 44 se mostrará el diagrama de Gantt con la duración de las tareas para crear la lección de "Repudiation Attack"

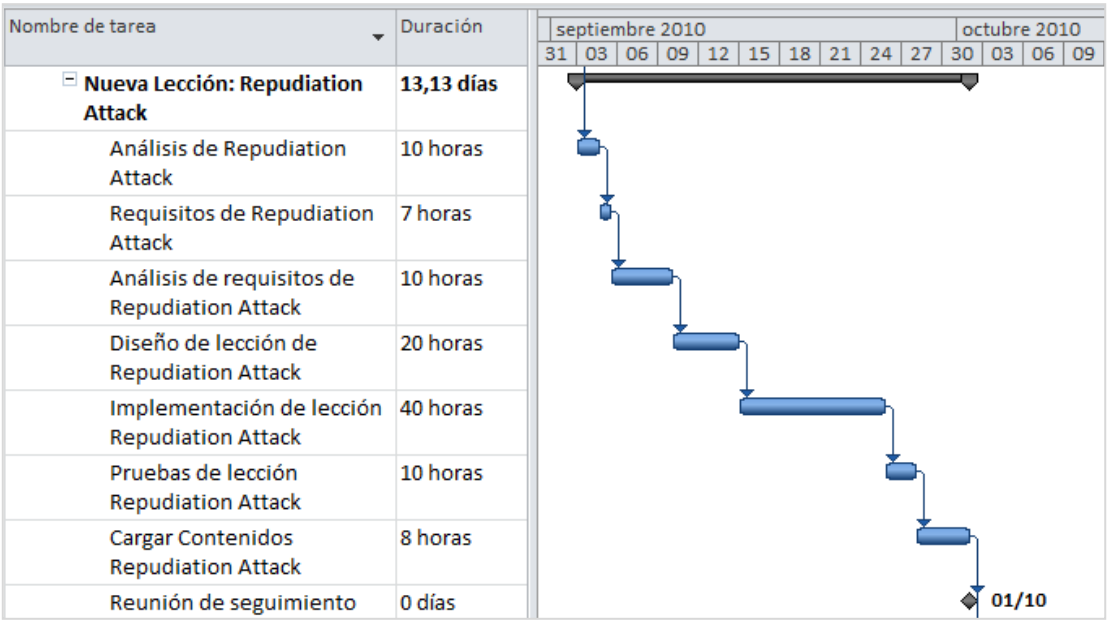

Ilustración 44 Diagrama de Gantt – Duración de tareas "Repudiation Attack"

Del diagrama de Gantt correspondiente a "Repudiation Attack", se obtendrán los resultados indicados en la Tabla 52:

Fecha de inicio de "Escaneo Anónimo de Puertos por CSPP"	02/09/2010
Fecha de finalización de "Escaneo Anónimo de Puertos por CSPP"	01/10/2010
Duración en días de "Escaneo Anónimo de Puertos por CSPP"	13,13 días
Horas totales empleadas	105 horas
Horas al día empleadas de lunes a viernes	2,5 horas / día
Horas al día empleadas los sábados	9 horas/día
Horas al día empleadas los domingos	4 horas/día

Tabla 52: Duración de tareas "Repudiation Attack"

7.1.3 Medios Técnicos Empleados

Esta sección contemplara todos los recursos técnicos empleados para la elaboración del proyecto.

Estos recursos se mostrarán a continuación en formato tabular, indicando nombre y descripción de cada uno.

7.1.3.1.1 Hardware

Nombre	Descripción
Ordenador portátil	SONY VAIO modelo VGN-FW11J: - Procesador Intel Core 2 DUO - 4 GB de memoria RAM
Ratón inalámbrico	Logitec
Router Inalámbrico	Comtrend

Tabla 53: Recursos Hardware

7.1.3.1.2 Telecomunicaciones

Nombre	Descripción
Conexión a Internet	ADSL de Movistar hasta 6 Mb de velocidad de bajada

Tabla 54: Recursos Telecomunicaciones

7.1.3.1.3 Sistema Operativo

Nombre	Descripción
M.S. Windows Vista	Windows Vista Home Premium de 32 bit con Service Pack 2
Linux	Ubuntu version 9.10.2
Máquina Virtual	Sun Virtual Box

Tabla 55: Recursos Sistema Operativo

7.1.3.1.4 Software para Desarrollo

Nombre	Descripción
Entorno de desarrollo	Eclipse versión 3.4.0
Plataforma	Java Platform (SE) versión 6.0.220.4
Librería XML para Java	SAX2
WebGoat	Código fuente de WebGoat, versión 5.2 para desarrolladores
Navegador Web	Mozilla Firefox versión 3.6.10

Tabla 56: Recursos Software Desarrollo

7.1.3.1.5 Software para Gestión y Documentación del Proyecto

Nombre	Descripción
Procesador de textos	Microsoft Word 2007
Gestión de Proyectos	Microsoft Project 2010
Software de Diagramado	Microsoft Visio Standard 2007

Tabla 57: Recursos Software Gestión y Documentación

7.2 Análisis Económico del Proyecto

Para analizar el coste económico del proyecto se tomará como referencia el apartado de "Planificación Inicial del Proyecto".

Las cantidades económicas indicadas de la Tabla 58 a la Tabla 65, tendrán el 18% de IVA incluido, exceptuando las donaciones a software libre.

7.2.1 Costes Iniciales

En este apartado se expondrán los recursos y personal que se estiman necesarios para el proyecto a priori.

7.2.1.1 Recursos

Los costes de los recursos materiales están basados en la sección "Medios Técnicos Empleados" y se detallarán en la Tabla 58

Concepto	Importe Total (IVA Incluido)	Vida estimada del producto	Horas de uso en el proyecto	Importe Proyecto (IVA incluido)
SONY VAIO modelo VGN-FW11J	1060,00 €	9125 horas[12]	780 horas	**90,60 €**
Ratón inalámbrico Logitec	30,00 €	4380 horas[13]	780 horas	**5,34 €**
Router Inalámbrico Comtrend	40,00 €	18250 horas[14]	300 horas	**0,65 €**

Tabla 58: Análisis Económico del Hardware Utilizado

El importe total contendrá el coste con IVA del producto en el momento de su adquisición.

La vida estimada del producto se corresponde con el número de horas en las que el producto pueda ser de utilidad.

El importe del proyecto es el resultado de multiplicar el precio por hora del producto por el número de horas que será empleado durante el proyecto.

Concepto	Importe Cuota Mensual (IVA Incluido)	Horas de uso en el proyecto	Importe Proyecto (IVA Incluido)
Conexión ADSL 6Mb + Teléfono	64,72€/mes	300 horas	**26,96 €**[15]

Tabla 59: Análisis Económico de Telecomunicaciones

[12] Estimación realizada tomando como base que el producto tendrá una duración de 5 años con una media de uso de 5 horas diarias

[13] Estimación realizada tomando como base que el producto tendrá una duración de 3 años con una media de uso de 3 horas diarias

[14] Estimación realizada tomando como base que el producto tendrá una duración de 5 años con una media de uso de 10 horas diarias

[15] Estimación realizada calculando el coste por hora, teniendo en cuenta meses de 30 días permaneciendo 24 horas conectado.

Concepto	Importe Total (IVA Incluido)	Vida estimada del producto	Horas de uso en el proyecto	Importe Proyecto (IVA Incluido)
Sistema Operativo Windows Vista Home Premium	124,32 €	9125 horas[16]	780 horas	**10,63 €**
Sistema Operativo Ubuntu	0 €	No aplica	No aplica	**0,00 €**
Máquina Virtual Sun Virtual Box	0 €	No aplica	No aplica	**0,00 €**

Tabla 60: Análisis Económico de Sistemas Operativos

El sistema operativo Ubuntu es de distribución libre, al igual que la máquina virtual de Sun Microsystem, Virtual Box.

Concepto	Importe Proyecto (IVA Incluido)
Eclipse versión 3.4.0	**0,00 €**
Java Platform (SE) versión 6.0.220.4	**0,00 €**
Librería Java XML SAX2	**0,00 €**
WebGoat, versión 5.2 para desarrolladores	**0,00 €**
Mozilla Firefox versión 3.6.10	**0,00 €**
Donación Software Libre	**100,00€**[17]

Tabla 61: Análisis Económico de Software de Desarrollo

Todo el software de desarrollo utilizado es gratuito, y por tanto no presentará coste alguno.

Se destinará parte de la inversión al software libre utilizado, debido a que sin sus aportaciones la realización de este proyecto hubiera supuesto un aumento de coste económico y de tiempo.

Concepto	Importe Total (IVA Incluido)	Vida estimada del producto	Horas de uso en el proyecto	Importe Proyecto (IVA Incluido)
Microsoft Word 2007	189,00 €	1825 horas	160 horas	**16,57 €**
Microsoft Visio Standard 2007	330,00 €	1825 horas	50 horas	**9,04 €**
Microsoft Project Standard 2010	775,00 €	1825 horas	30 horas	**12,74 €**

Tabla 62: Análisis Económico de Software de Gestión y Documentación

Las estimaciones de costes para el software de gestión y documentación se han realizado en base al tiempo estimado para la elaboración de la documentación del proyecto (160 horas).

[16] Estimación realizada tomando como base que el producto tendrá una duración de 5 años con una media de uso de 5 horas diarias

[17] Las donaciones no requieren IVA, este es el único caso en el que no se incluye.

El valor estimado de los productos de software de gestión y documentación se han calculado en base a una estimación de vida de 5 años, contando con una media de uso diario de una hora.

Además de los gastos indicados, existen una serie de gastos generales que también deberán tomarse en cuenta:

Concepto	Precio Unitario	Cantidad	Importe Total (IVA Incluido)
Cuotas Seguridad Social	250,86 €/mes	8 meses	2006,88 €
Seguro de Responsabilidad Civil	46,51 €/mes	8 meses	372,08 €
Seguro de Autónomo	1,62 €/mes	8 meses	12,96 €
Alquiler Oficina	100 €/mes	8 meses	800,00 €
Energía Eléctrica Consumida	6 €/mes	8 meses	48,00 €
Agua	6 €/mes	8 meses	48,00 €
Gas	6 €/mes	8 meses	48,00 €
Transporte	-	-	30,00 €
Papelería	-	-	10,00 €

Tabla 63: Análisis Económico de Gastos Generales

La cuota de la seguridad social (Ministerio de Trabajo e Inmigración, Gobierno de España) será calculada en base a lo establecido por esta entidad para el año 2010. Se cotizará con la base mínima (841,80 €), que se corresponderá con un tipo del 29,80%, por lo cual se pagará una cuota mensual de 250,86 €/mes.

El seguro de responsabilidad civil se corresponde con el precio establecido por la entidad aseguradora Alfa - Risk (Alfa-Risk SL Correduría de Seguros) para informáticos autónomos.

El seguro de autónomo será contratado a la entidad aseguradora Mapfre (MAPFRE INTERNET, S.A.).

Para estimar la energía eléctrica consumida, se tomará como referencia la factura mínima de luz emitida por Iberdrola (aproximadamente 6 € mensuales) y se multiplica por el número de meses de duración del proyecto.

7.2.1.1.1 Coste Total Inicial de Recursos

Sumando todos los gastos ocasionados por los recursos, se estimará que **el coste sería de 3.648,45 €** en los 8 meses en los que se estima su duración.

En la Tabla 64 se detallarán estos costes.

Concepto	Importe Total (IVA Incluido)
Hardware	96,59 €
Telecomunicaciones	26,96 €
Sistemas Operativos	10,63 €
Software de Desarrollo	100,00 €
Software de Gestión y Documentación	38,35 €
Gastos Generales	3.375,92 €
TOTAL	**3.648,45 €**

Tabla 64: Coste Total Inicial de Recursos

7.2.1.2 Personal

A continuación se analizarán los costes de personal, que se corresponderán con el importe asociado el tiempo invertido por un Ingeniero Técnico Informático para la elaboración integra del proyecto.

Realizará labores de jefe de proyecto, analista, diseñador, programador e ingeniero de pruebas, además de la presente documentación.

La unidad de referencia para medir el trabajo del recurso serán las horas empleadas. Por tanto, para realizar el cálculo de costes del recurso, multiplicaremos el coste por hora del recurso por el número de horas empleadas.

Se estimará el coste por hora del recurso en 35 €/hora (IVA no Incluido). Obteniendo los datos indicados en la Tabla 65:

Recurso	Número de Recursos	Coste €/hora (IVA Incluido)	Horas Invertidas	Total (IVA Incluido)
Ingeniero Técnico en Informática	1	41,30 €/hora	780 horas	**32.214,00 €**

Tabla 65: Análisis Económico de Recursos Humanos

7.2.2 Presupuesto Inicial

Para realizar la primera estimación de costes habrá que partir de la estimación inicial de tiempos explicada en el apartado de "Planificación Inicial" y en los costes de los medios técnicos que en un principio se estimaron como necesarios.

El proyecto se cuantificará en horas, por lo que se establecerá un coste por hora que cubra todos los costes del proyecto. Este coste vendrá determinado por el coste por hora de recursos más el coste por hora de personal:

Coste Recursos (IVA Incluido)	Coste Personal (IVA Incluido)	Coste por hora (IVA Incluido)
41,30 €/hora		**45,98 €**

Tabla 66: Coste por hora del Proyecto

En la Tabla 67 se detallará el presupuesto del proyecto planteado inicialmente[18]. El importe de cada concepto se corresponde con el coste por hora del proyecto (calculado en la Tabla 66) por el número de horas indicado:

CONCEPTO	CANTIDAD	IMPORTE (IVA Incluido)
Análisis Inicial	50 horas	2.299,00 €
Sistema Multilenguaje	176 horas	8.092,48 €
Nuevas Lecciones:		
- Análisis previo	116 horas	5.333,68 €
- Clickjacking	108 horas	4.965,84 €
- Escaneo Anónimo de Puertos por CSPP	108 horas	4.965,84 €
- Repudiation Attack	62 horas	2.850,70 €
Documentación	160 horas	7.356,80 €
	TOTAL	35.864,34 €

Tabla 67: Análisis Económico Presupuesto Inicial

7.2.3 Presupuesto para el Cliente

A partir del apartado anterior se elaborará el presupuesto que se deberá entregar al cliente para informar de los costes del proyecto.

Además se tendrán en cuenta dos factores:

1. Porcentaje de Beneficio: para que el proyecto sea rentable. Se aplicará un 18% de beneficio.
2. Porcentaje de Riesgo: se sumará un 10% para prever posibles inconvenientes que puedan surgir durante la elaboración del proyecto y que supongan un gasto no contemplado en el análisis inicial de costes. En este caso los factores de riesgo serían dos:
    a. Desconocimiento de la plataforma de desarrollo y lenguaje de programación. El aprendizaje de este entorno podría retrasar los plazos estimados inicialmente.
    b. Uso de código heredado. Pueden encontrarse ciertas partes de código que sean difíciles de comprender y requieran un esfuerzo extra.

El presupuesto que se entregará al cliente tendrá un incremento de un 28% (18% de beneficios y 10% de riesgo)

[18] Todos los importes que aparecerán en el presupuesto tendrán el IVA incluido.

CONCEPTO	CANTIDAD	IMPORTE (IVA Incluido)
Análisis Inicial	50 horas	2.942,72 €
Sistema Multilenguaje	176 horas	10.358,37 €
Nuevas Lecciones:		
- Análisis previo	116 horas	6.827,11 €
- Clickjacking	108 horas	6.356,27 €
- Escaneo Anónimo de Puertos por CSPP	108 horas	6.356,27 €
- Repudiation Attack	62 horas	3.648,90 €
Documentación	160 horas	9.416,70 €
	TOTAL	45.906,34 €

Tabla 68: Análisis Económico Presupuesto para el Cliente

7.2.4 Coste Final y Análisis de la Desviación

Para analizar el coste económico final del proyecto se tomará como referencia el apartado "Desarrollo Real del Proyecto".

Las cantidades económicas indicadas de la Tabla 69 a la Tabla 75, tendrán el 18% de IVA incluido.

7.2.4.1 Costes Reales

A continuación se expondrán los costes reales del proyecto una vez finalizado. Se han tenido en cuenta los recursos y personal utilizado para la elaboración del proyecto para el cálculo de los costes.

7.2.4.1.1 Recursos

Los costes de los recursos materiales están basados en la sección "Medios Técnicos Empleados" y se detallarán en la Tabla 69 con la duración real del proyecto

Concepto	Importe Total (IVA Incluido)	Vida estimada del producto	Horas de uso en el proyecto	Importe Proyecto (IVA incluido)
SONY VAIO modelo VGN-FW11J	1060,00 €	9125 horas[19]	1108 horas	128,71 €
Ratón inalámbrico Logitec	30,00 €	4380 horas[20]	1108 horas	7,59 €
Router Inalámbrico Comtrend	40,00 €	18250 horas[21]	500 horas	1,10 €

Tabla 69: Coste Real del Hardware Utilizado

En el importe total se incluirá el coste con IVA del producto en el momento de su adquisición.

[19] Estimación realizada tomando como base que el producto tendrá una duración de 5 años con una media de uso de 5 horas diarias
[20] Estimación realizada tomando como base que el producto tendrá una duración de 3 años con una media de uso de 3 horas diarias
[21] Estimación realizada tomando como base que el producto tendrá una duración de 5 años con una media de uso de 10 horas diarias

La vida estimada del producto se corresponderá con el número de horas en las que el producto pueda ser de utilidad.

El importe del proyecto es el resultado de multiplicar el precio por hora del producto por el número de horas que será empleado durante el proyecto.

Concepto	Importe Cuota Mensual (IVA Incluido)	Horas de uso en el proyecto	Importe Proyecto (IVA Incluido)
Conexión ADSL 6Mb + Teléfono	64,72€/mes	500 horas	**44,94 €**[22]

Tabla 70: Coste Real de Telecomunicaciones

Concepto	Importe Total (IVA Incluido)	Vida estimada del producto	Horas de uso en el proyecto	Importe Proyecto (IVA Incluido)
Sistema Operativo Windows Vista Home Premium	124,32 €	9125 horas[23]	1108 horas	**15,10 €**
Sistema Operativo Ubuntu	0 €	No aplica	No aplica	**0,00 €**
Máquina Virtual Sun Virtual Box	0 €	No aplica	No aplica	**0,00 €**

Tabla 71: Coste Real de Sistemas Operativos

Concepto	Importe Proyecto (IVA Incluido)
Eclipse versión 3.4.0	**0,00 €**
Java Platform (SE) versión 6.0.220.4	**0,00 €**
Librería Java XML SAX2	**0,00 €**
WebGoat, versión 5.2 para desarrolladores	**0,00 €**
Mozilla Firefox versión 3.6.10	**0,00 €**
Donación Software Libre	**100,00€**[24]

Tabla 72: Coste Real de Software de Desarrollo

Se mantendrá el importe de donación a software libre; no se tendrá en cuenta la desviación de tiempo en este caso.

Concepto	Importe Total (IVA Incluido)	Vida estimada del producto	Horas de uso en el proyecto	Importe Proyecto (IVA Incluido)
Microsoft Word 2007	189,00 €	1825 horas	230 horas	**23,81 €**
Microsoft Visio Standard 2007	330,00 €	1825 horas	90 horas	**16,27 €**
Microsoft Project Standard 2010	775,00 €	1825 horas	60 horas	**25,48 €**

[22] Estimación realizada calculando el coste por hora, teniendo en cuenta meses de 30 días permaneciendo 24 horas conectado.
[23] Estimación realizada tomando como base que el producto tendrá una duración de 5 años con una media de uso de 5 horas diarias
[24] Las donaciones no requieren IVA, este es el único caso en el que no se incluye.

Tabla 73: Coste Real de Software de Gestión y Documentación

Las estimaciones de costes para el software de gestión y documentación se han realizado en base al tiempo empleado en la elaboración de la documentación del proyecto (230 horas).

El valor estimado de los productos de software de gestión y documentación se han calculado en base a una estimación de vida de 5 años, contando con una media de uso diario de una hora.

Los gastos generales del proyecto se detallarán en la Tabla 74:

Concepto	Precio Unitario	Cantidad	Importe Total (IVA Incluido)
Cuotas Seguridad Social	250,86 €/mes	12 meses	3.010,32 €
Seguro de Responsabilidad Civil	46,51 €/mes	12 meses	558,12 €
Seguro de Autónomo	1,62 €/mes	12 meses	19,44 €
Alquiler Oficina	100 €/mes	12 meses	1.200,00 €
Energía Eléctrica Consumida	6 €/mes	12 meses	72,00 €
Agua	6 €/mes	12 meses	72,00 €
Gas	6 €/mes	12 meses	72,00 €
Transporte	-	-	30,00 €
Papelería	-	-	10,00 €

Tabla 74: Coste Real de Gastos Generales

7.2.4.1.1.1 Coste Total Inicial de Recursos

Sumando todos los gastos ocasionados por los recursos, se estimará que **el coste sería de 5.406,88 €** en los 12 meses en los que se estima su duración.

Concepto	Importe Total (IVA Incluido)
Hardware	137,40 €
Telecomunicaciones	44,94 €
Sistemas Operativos	15,10 €
Software de Desarrollo	100,00 €
Software de Gestión y Documentación	65,56 €
Gastos Generales	5.043,88 €
TOTAL	**5.406,88 €**

7.2.4.1.2 Personal

Los costes ocasionados por el personal aumentarán proporcionalmente con el número de horas que se han tenido que invertir en el proyecto

Recurso	Número de Recursos	Coste €/hora (IVA Incluido)	Horas Invertidas	Total (IVA Incluido)
Ingeniero Técnico en Informática	1	41,30 €/hora	1108	45.760,40 €

	horas

Tabla 75: Coste Final de Recursos Humanos

7.2.4.2 Análisis del Beneficio Neto

Para realizar el cálculo del beneficio neto real que da el proyecto deberemos restar al importe de la factura, los gastos producidos por la declaración de IVA y el IRPF, que serán calculados en las siguientes sub-secciones

7.2.4.2.1 Declaración de IVA

Para calcular el importe de IVA que se deberá abonar, se restará el IVA imputado al cliente menos el IVA soportado por los costes ocasionados durante la elaboración del proyecto.

El IVA imputado se calcula a partir del presupuesto presentado al cliente, como se muestra en la Tabla 76:

Importe Neto Presupuesto	% IVA	Importe Bruto Presupuesto	IVA imputado
	18 %	45.906,34 €	**7.002,66 €**

Tabla 76: IVA Imputado

La Tabla 77 contendrá los datos del IVA soportado:

Concepto	Importe IVA
Conexión ADSL + Teléfono	6,86 €
Seguro Responsabilidad Civil	85,14 €
Seguro Autónomo	2,97 €
Alquiler Oficina	183,05 €
Energía Eléctrica	10,98 €
Agua	10,98 €
Gas	10,98 €
Trasporte	4,58 €
Papelería	1,52 €
TOTAL	**317,06 €**

Tabla 77: Cálculo IVA Soportado

El pago de IVA que se deberá realizar se muestra en la Tabla 78

IVA Imputado	IVA Soportado	IVA a declarar
	317,06 €	**6.685,60 €**

Tabla 78: Coste Final Declaración de IVA

7.2.4.2.2 Declaración del IRPF

La declaración de IRPF se mantendrá igual que en el análisis previo como muestra la Tabla 79

Total Factura	Porcentaje IRPF	IRPF a declarar
	15%	**6.885,95 €**

Tabla 79: Coste Final Declaración de IRPF

7.2.4.2.3 Cálculo del Beneficio Neto

Partiendo de los datos obtenidos en las sub-secciones anteriores se calcula beneficio neto en la Tabla 80:

Total Factura	Costes Proyecto	Importe IRPF	Importe IVA	Beneficio Neto
	- 51.167,28 €	- 6.885,95 €	-6.685,60 €	**-18.832,49 €**

Tabla 80: Coste Final Cálculo del Beneficio Neto

7.2.4.3 Análisis de la Desviación

Al finalizar el proyecto se obtienen las siguientes conclusiones:

1. El proyecto no ha resultado rentable; presenta un déficit de 18.832,49 €.
2. No se han cumplido las estimaciones iniciales de tiempo.
3. Los costes del proyecto han aumentado en 15.304,83€ (un 42,68% más del presupuesto estimado inicialmente).
4. El aumento del coste está directamente relacionado con el incremento de tiempo que ha requerido el proyecto para su realización.

Los factores que han causado el incremento del tiempo para la elaboración del proyecto son:

1. Poca experiencia en el entorno de trabajo (plataforma de desarrollo y lenguaje de programación)
2. El tiempo empleado en comprender el funcionamiento del código existente
3. Rendimiento del recurso inferior al esperado.

8 ANEXO II: Glosario de Términos

A

Aplicación Web: Programa preparado para una utilización específica, sujeto a una estructura cliente – servidor, donde el medio de comunicación puede ser Internet o una intranet, y el usuario visualiza la información a través de un navegador.

C

Caso de Uso: Describe una funcionalidad que un usuario tiene en una aplicación

Clase: En programación orientada a objetos, es una entidad abstracta que sirve como modelo para crear objetos

Clickjacking: Tipo de ataque sobre aplicaciones Web, que consiste en engañar al usuario para que realice determinadas acciones creyendo que está realizando otras

Cliente: En informática se refiere una entidad que solicita información.

Cookies: En el ámbito informático se refiere a información del usuario cliente almacenada que será utilizada por el servidor

CSPP: Connection String Parameter Pollution, es un tipo de ataque realizado sobre aplicaciones que se conectan a bases de datos utilizando cadenas de conexión.

CSS: Cascade Style Sheet. Lenguaje usado para definir

D

Diagrama de Gantt: Es una representación gráfica que muestra el tiempo dedicado a cada tarea de un proyecto y la secuencia en la que se realizan

Diagrama de Secuencia: Muestra la interacción que debe darse entre objetos a través del tiempo para desarrollar una acción dentro de una aplicación.

H

HSQLDB: Hyper Structured Query Language Data Base. Gestor de base de datos implementado en Java.

HTML: HiperText Markup Language, Lenguaje de marcado utilizado para la elaboración de páginas Web.

I

IRPF: Impuesto sobre la Renta de las Personas Físicas, Pago que se debe realizar a la Agencia Tributaria al emitir una factura por una actividad comercial realizada.

IVA: Impuesto de Valor Añadido, es un tributo de naturaleza indirecta que recae sobre el consumo y grava: las entregas de bienes y prestaciones de servicios efectuadas por empresarios y profesionales, las adquisiciones intracomunitarias y las importaciones de bienes.

Iframe: Elemento HTML que permite definir un área dentro de la pantalla del navegador, en la cual se puede cargar una página web.

J

J2EE: Java 2 Enterprise Edition, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java.

Java: Lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems.

Javascript: Lenguaje de programación utilizado por aplicaciones Web para ejecutar scripts en el equipo cliente.

N

Navegador: Aplicación instalada en los equipos cliente, a través de la cual el usuario podrá ver e interactuar con la información enviada por el servidor Web

O

Objeto: En programación orientada a objetos, es una entidad concreta creada a partir de una clase que definirá sus propiedades y métodos.

OWASP: Open Web Application Security Project, es una comunidad formada para mejorar la seguridad en aplicaciones software.

R

RAE: Real Academia Española (http://www.rae.es)

S

SAX: Librería Java para interpretar documentos XML.

Servidor: En informática se refiere a la entidad que recibe y procesa las peticiones del cliente, enviando una respuesta si procede**.**

W

WebGoat: es una aplicación web J2EE deliberadamente insegura, mantenida por OWASP y diseñada para enseñar lecciones de seguridad en aplicaciones Web.

X

XML: Extensible Markup Language, es un metalenguaje basado en etiquetas que permite crear estructuras para almacenar o intercambiar información.